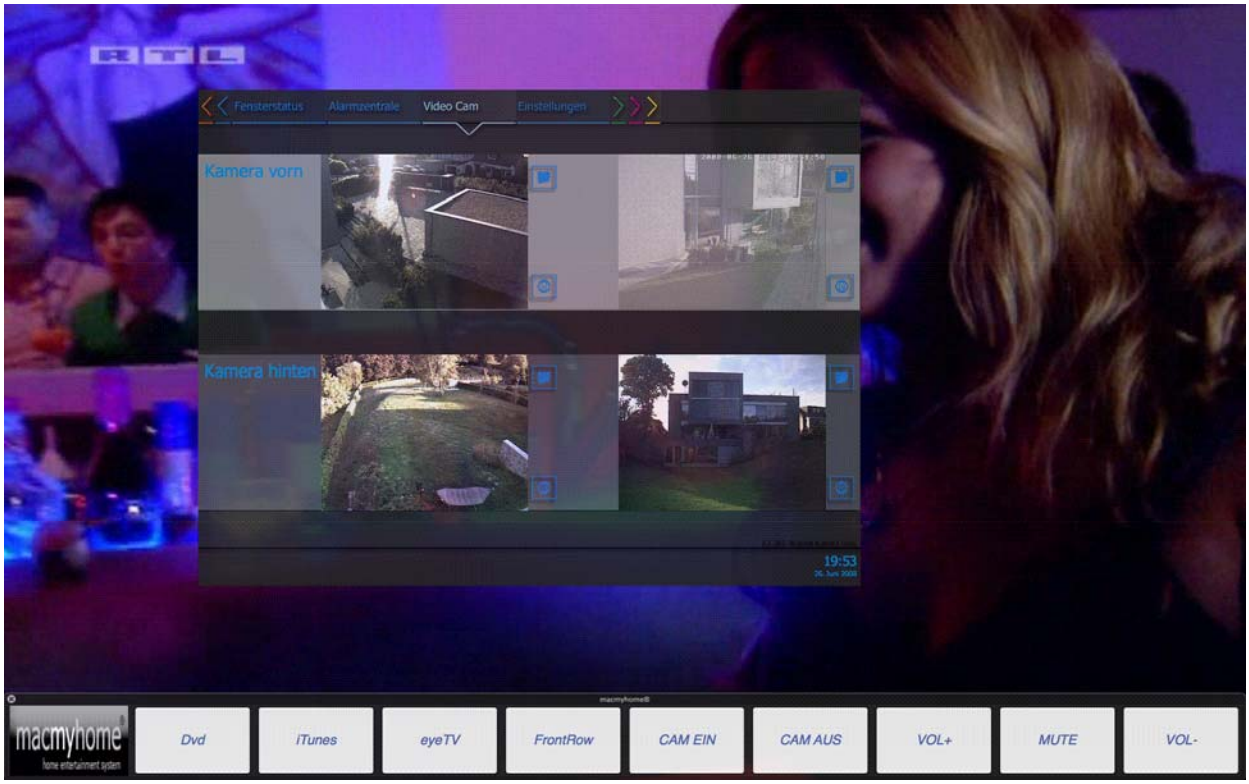
ibs
intelligent building services gmbh

m..myhome

Version 1.1.RC3, Stand: November 2009

Bedienungs- und Benutzerhandbuch für Systemintegratoren



m..myhome *for Mac*

home entertainment system

1. Einleitung **4**
2. Installation **4**
3. Konfiguration, Lizenzierung und Steuerung **5**
3.2 Lizenzierung 8
3.2.1 Einspielen einer Lizenzdatei 9
3.2.2 Anfordern einer Lizenzdatei 10
3.3 Steuerung des m..myhome - Dienstes 10
3.3.1 Aktivieren des Dienstes 10
3.3.2 Beenden des Dienstes 10
3.3.3 Statusanzeige 10
3.3.4 Logging Monitor 10
3.3.5 Die m..myhome Verzeichnisstruktur 12
3.3.6 Die m..myhome Firewall 13
3.4. Scripting Client Tool 15
3.4.1 Installation 15
3.4.2 Installation der LabVIEW Runtime Umgebung 16
3.4.3 Anwendung 17
3.4.4 Umgang mit dem Scripting Client 21
3.4.5 Monitoring 27
3.4.6 Testfeld 28
3.4.7 Weitere Funktionen 29
4. Der m..myhome Befehlssatz **29**
4.1 Begrifflichkeiten / Typen 29
4.1.1 Klassen (CLASS) 29
4.1.2 Kommandos (CMD) 30
4.1.3 Scripte 30
4.1.4 Sonderkommandos 30
4.1.5 Formatierungsoptionen (optional) 30
4.1.6 Konverter bzw Konvertierungsoptionen (optional) 30
4.2 Syntax (Aufbau des Protokolls) 30
4.2.1 Iterative Abarbeitung 32
4.3 Antwortsequenz 32
4.4 Formatieroptionen, Parser Optionen, Status Port 33
4.4.1 Formatieroptionen. 33
4.4.2 Konvertieroptionen. 36
4.4.3 Parser Optionen (Match Bedingungen). 38
4.4.4 Argumente (Übergabe). 38
4.4.5 Automatisch generierte Ausgaben (Statusport). 39
4.5 Befehlsklassen 40
4.6 Kommandolisten 41
4.6.1 BROWSER 41
4.6.2 DVD 42
4.6.3 ITUNES 44
4.6.4 SLIDE 46
4.6.5 SCRIPT 47
4.6.6 SCRIPTLOAD 47
4.6.7 SYS 48
4.6.8 TV 52
4.6.9 VLC 54
4.6.10 UDP/TCP 55
4.6.11 Window 56
4.6.12 BAOS 58
4.6.13 HS 58
4.6.14 Streamer 59
4.6.15 KNX 59
4.6.16 TIMER 59

4.6.17 Sonderkommandos 60
4.7 Tastaturbelegung (Sondertasten) 60
5. Erweiterte Funktionen (AddOn's) 61
5.1. Die CommandServer Erweiterung 61
5.1.1 Aufbau der CommandServer Definitionstabellen 63
5.1.2 Reservierte (extended) Servertypen [SERVERTYPE] 73
5.1.3 Verwendung von USB/RS232 Umsetzern 74
5.2. Slimserver, xPL 76
5.2.1. Konfiguration der xPL Klasse 77
5.3. Event Server 79
5.4. Die GUI Buttonleisten 82
5.4.2 Extended Button Leisten 84
5.5. GIRA Homserver KO Gateway, KNXnet/IP Support 89
5.5.1 KNX IP BAOS 770 89
5.5.2 GIRA Homeserver KO-Gateway Support 92
5.5.3 KNXnet/IP (Routing und Tunneling) 94
5.5.3.1 KNXnet/IP Routing Support 94
5.5.3.2 KNXnet/IP Tunneling Support 96
5.5.3.3 KNXnet/IP TUNNELING_BRIDGE Support 97
5.6. Airport Support (Airfoil) 99
5.7. eyeTV - Streaming Support 101
5.8. m..Remote Support (cf iViewer) 103
6. Sonstiges 104
6.0. Hinweise zur Konfiguration Ihres Mac 104
6.1. Verwendung von Applescript 104
6.2. Arbeiten mit Systemvariablen 105
6.3 Mini- Webserver 107
6.4 Timer Support 109
6.5 Counter Support 111
7. TABELLE key codes SYSTEM EVENTS 113
8. Workarounds, Previews 114
8.1. DVD-Copy-Window 114
8.2. iTunes Remote 114
8.3. Nutzung der Squeeze Box als IR Empfänger zur Steuerung von m..myhome 116
8.4. Nützliche Shell Kommandos 117
8.5. Nützliche Shortcuts und Tastaturbefehle 118
8.6. Sonderzeichen (Mac/Win/VM-Ware) 119
10. Versions Historie 121

Sehr geehrter m..myhome Kunde,

wir gratulieren zum Erwerb dieses ibs-Produktes. ibs ist bestrebt bei seinen Produkten höchste Qualitäts- und Verarbeitungsstandards einzuhalten, um Ihnen ein optimales Preis-Leistungsverhältnis zu bieten. Sollten Sie trotzdem einmal Probleme mit dem Produkt oder Anregungen für Produktverbesserungen haben, setzen Sie sich bitte direkt mit uns in Verbindung. Dieses Handbuch ist für Systemintegratoren geschrieben. Entsprechende Kenntnisse setzen wir voraus.

Wir wünschen Ihnen viel Freude bei der Anwendung.

1. Einleitung

Die **m..myhome** - Software stellt als Hintergrundprozess einen TCP-/UDP-Server-Dienst zur Steuerung eines Apple Mac – Systems (Mac mini bevorzugt) per Netzwerk zur Verfügung. Ein speziell für diese Anwendung definiertes Protokoll ermöglicht es, über eine einfache Netzwerkanbindung per LAN oder WLAN vielfältige Aktionen am Mac auszuführen und diesen umfangreich zu kontrollieren. Bitte beachten Sie, dass die Abbildungen teilweise nicht den aktuellen Softwarestand aufzeigen, können also leicht abweichen.

2. Installation

Sie starten die Installation von **m..myhome** durch Doppelklick auf das **m..myhome**-Paket:

Sie werden daraufhin durch die Installation geführt:

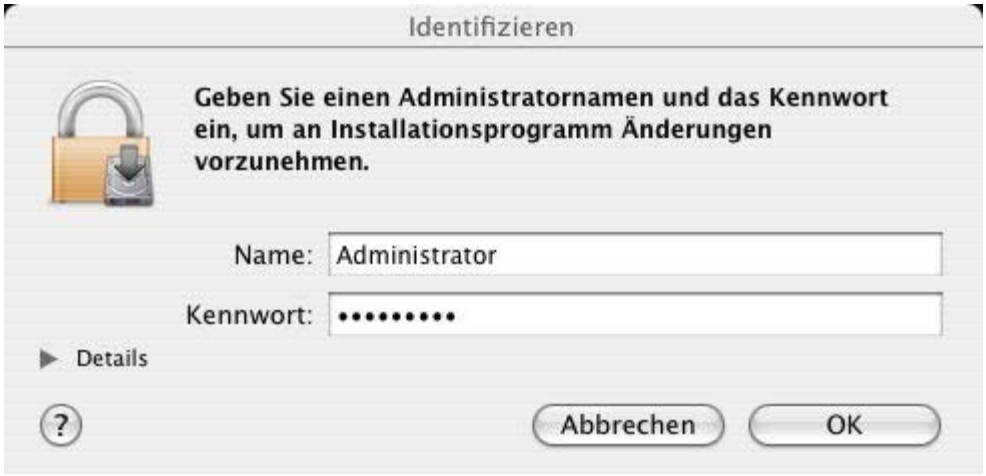
„MacMyHome" installieren
Willkommen bei: MacMyHome
Einführung
Lizenz
Zielvolume auswählen
Installationstyp
Installation
Zusammenfassung
Herzlich Willkommen zur Installation der m..myhome - Software für Mac OS X.
Sie werden nun durch die Installation geleitet.
Bitte beachten Sie, dass zur Nutzung der Software eine gültige Lizenzdatei erforderlich ist. Ohne gültige Lizenzdatei startet die Software im Demo Modus.
Zurück
Fortfahren

Klicken Sie auf „Fortfahren".

Lesen Sie die erscheinende Lizenzvereinbarung aufmerksam durch und klicken Sie erneut auf „Fortfahren". Bestätigen Sie die nachfolgende Abfrage, wenn Sie der Lizenz zustimmen, durch einen

Klick auf „Agree". Das Installationsprogramm zeigt Ihnen im Anschluss daran an, dass **m..myhome** auf der Systempartition installiert wird. Eine Installation auf einer anderen Partition ist systembedingt nicht möglich. Bestätigen Sie dies durch einen Klick auf „Fortfahren". In dem nun erscheinenden Fenster klicken sie bitte auf „Aktualisieren", um die Installation durchzuführen.

Zur Installation von **m..myhome** sind Administratorrechte erforderlich. Bitte geben Sie die erforderlichen Daten ein:

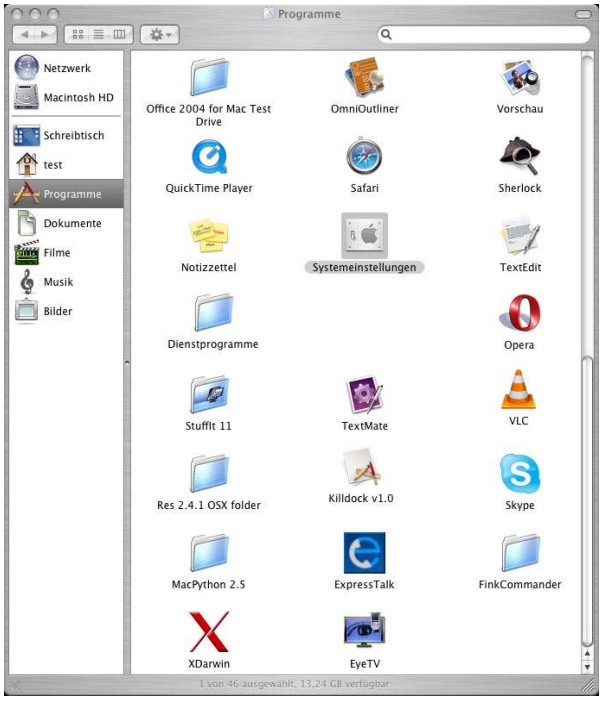


Bestätigen Sie Ihre Eingaben mit „OK". **m..myhome** wird nun installiert. Bestätigen Sie den Abschluss der Installation durch einen Klick auf „Schließen".

Herzlichen Glückwunsch! **m..myhome** ist nun auf Ihrem System installiert. Bitte beachten Sie, dass die Installation von **m..myhome** immer benutzerbezogen stattfindet. Eine Mehrfachinstallation von **m..myhome** unter verschiedenen Benutzern auf einem System ist nicht zu empfehlen, da dies u.U. Zu Funktionsstörungen führen kann.

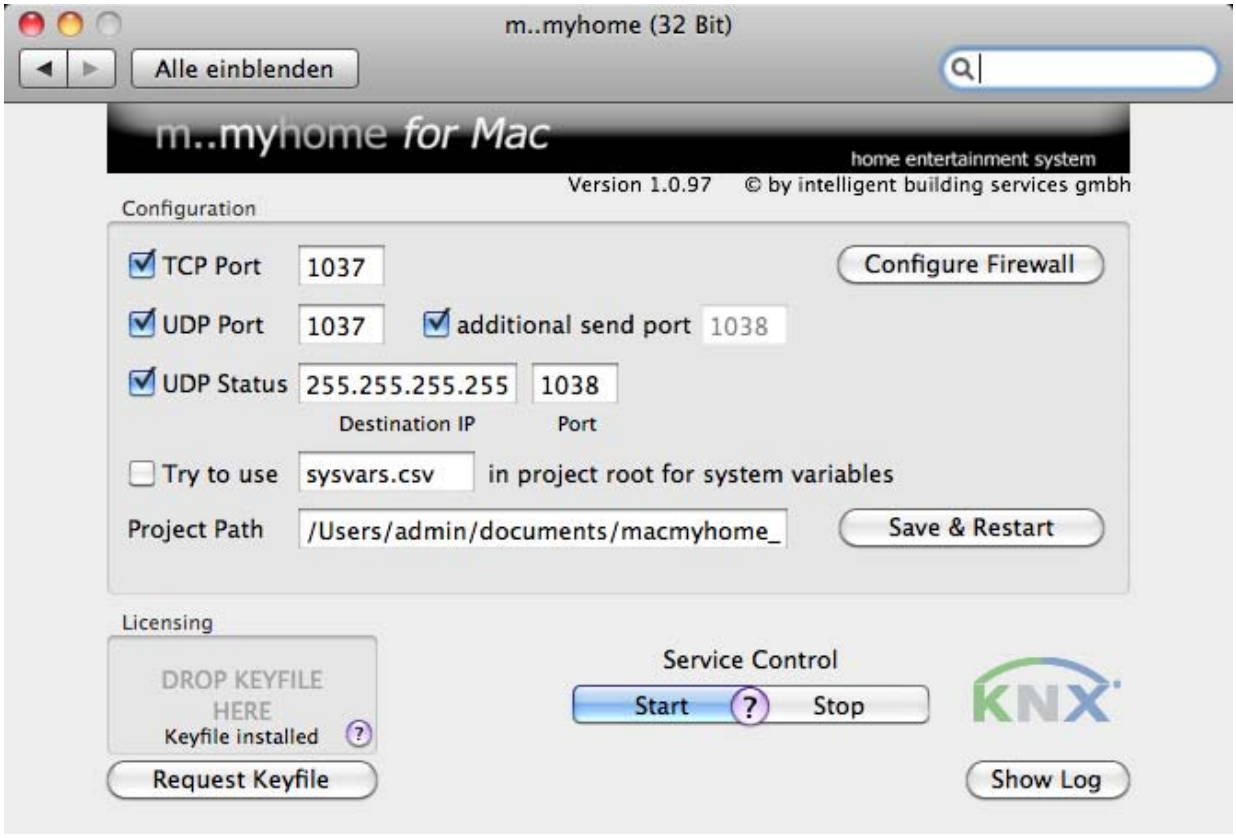
3. Konfiguration, Lizenzierung und Steuerung

Zur Konfiguration von **m..myhome** öffnen Sie bitte die Finder-Applikation, klicken im linken Fensterteil auf „Programme" und wählen anschließend „Systemeinstellungen":



Doppelklicken Sie auf „Systemeinstellungen". Im nun erscheinenden Fenster klicken Sie bitte unter „Sonstige" auf das **m..myhome**-Icon:

Sie befinden sich nun in der Konfiguration von **m..myhome**:



3.1 Konfiguration

Im Feld „Configuration" werden alle zum Betrieb des **m..myhome** erforderlichen Netzwerkeinstellungen vorgenommen. Änderungen an der Konfiguration werden durch einen Klick auf „Save & Restart" dauerhaft abgespeichert und der Dienst ggfs. neu gestartet, so dass Änderungen an der Konfiguration sofort übernommen werden. **m..myhome** kann sowohl TCP- als auch UDP-Verbindungen annehmen.

Die einzelnen Dienste werden durch Setzen eines Hakens aktiviert.

TCP Port: Gibt den Port an, auf dem **m..myhome** auf eingehende TCP-Verbindungen wartet. Gültiger Wertebereich: 1024 – 1038 und 1040 – 65535.

Hinweis:

Eine bestehende Verbindung auf einem TCP Port hat sperrende Wirkung, so dass während dessen keine UDP-Kommandos verarbeitet werden.

UDP Port:
Gibt den Port an, auf dem **m..myhome** auf UDP-Pakete wartet. Gültiger Wertebereich: 1024 – 65535

additional send port:
Durch Aktivieren dieser Option aktivieren Sie das zusätzliche Senden der Antwortsequenz von **m..myhome** an den Client auf einem zusätzlichen Port. Diese Option ist als Kompatibilitätsmodus für Clients mit eingeschränkter Netzwerkfunktionalität zu verstehen und sollte nur im Bedarfsfall aktiviert werden. Der zusätzliche Send-Port entspricht immer dem gewählten UDP Port+1.

UDP Status:
Geben Sie hier die IP (Broadcastadressen sind erlaubt) und den Port an, an den Statusinformationen über das System gesendet werden. **m..myhome** sendet die Statusinformationen automatisch bei Änderungen, ohne dass ein Client diese Informationen explizit abfragen muss.

Hinweis:

Der TCP Port 1039 ist für Debug- und Logging-Funktionen reserviert und deshalb nicht auswählbar. Um diese Funktion zu nutzen, bauen Sie bitte eine zusätzliche TCP-Verbindung zu* m..myhome** *auf diesem Port auf.* **m..myhome *wird Ihnen daraufhin zeitnah Informationen über die Abarbeitung von empfangenen Kommandos senden. Es können bis zu 16 Clients gleichzeitig auf den Port 1039 zugreifen.

3.2 Lizenzierung

Um **m..myhome** in vollem Umfang nutzen zu können, benötigen Sie eine gerätebezogene Lizenzdatei. Ohne Lizenz wird **m..myhome** nur im Demo-Modus ausgeführt. Im Demo-Modus sind nur eingeschränkte Möglichkeiten verfügbar. Darüber hinaus wird alle 20min eine Meldung generiert sowie die Tonausgabe abgestellt (Mute).

m..myhome ist modular aufgebaut und kann beliebig erweitert werden. Das Programmpaket besteht aus einer Basislizenz und den entsprechenden, optionalen Erweiterungen. Zurzeit stehen folgende Module und Systemschnittstellen zur Verfügung:

Basislizenzen:

m..myhome Gate	*Grundversion, **nicht erweiterbar**, für Anwender die das Mac System z.B als einfachen Audiozuspieler verwenden wollen. Beinhaltet einen CommandServer, WebServer, einfache Buttonleisten (Standard GUI). Diese Version ist auch zur Steuerung eines einfachen Medicenters (Frontrow, eyeTV) geeignet.*
m..myhome Gate *plus*	*Grundversion, wie vor, jedoch erweiterbar und beinhaltet 10 CommandServer (erweiterbar bis auf max 25).*

Erweiterungen / AddOn's (nur in Verbindung mit der Gate *plus* Lizenz):

CommandServer AddOn	*Erweiterung der CommandServer Schnittstelle um jeweils 5 Protokolle (Es sind max. 25 Protokolle je Mac System möglich)*
XPL AddOn	*Einbindung des xPL Protokolls für zB die Ansteuerung der Logitech Squeeze Box.*
EventServer AddOn	*Globale Schnittstelle zum Empfangen und Auswerten diverser TCP/UDP Protokolle. (TCP/UDP Listener)*
Extended Buttons AddOn	*Erweiterte Buttonleisten Steuerung. (max. 25 Buttonleisten definierbar. Erweiterter Befehlssatz für die Einrichtung der Buttons.*
BAOS / KNX AddOn	*Modul für die Einbindung des Weinzierl KNX IP 770 Interface (Objektserver), und/oder GIRA Homeserver KO-Gateway, und/oder KNXnet/IP (Routing oder Tunneling).*
Streaming AddOn	*Modul für den eyeTV Streaming Service. Streamt den aktuellen Transponder ins Netzwerk und kann zb über VLC empfangen werden.* ***(Dieses Modul befindet sich in der Testphase und wird z.Z kostenlos und ohne Anspruch auf Vollständigkeit***

ausgeliefert.)

Systemschnittstellen:

m..Remote Server AddOn	*Kommunikationsserver für die Einbindung der m..Remote Steuerung (iPhone / iPod Touch Visualisierung). Dieses Modul wird je Mac System, auf dem per iPhone / iPod Touch zugegriffen werden soll, empfohlen.* ***Hinweis:*** *Es ist lediglich ein m..Remote Server AddOn je Infrastruktur notwendig. Die Umsetzung mit nur einem m..Remote Server erfordert jedoch einen erheblich höheren Konfigurations-aufwand.ausgeliefert.)*

Client Lizenzen:

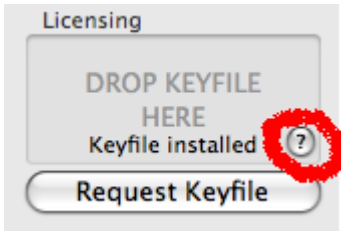
m..Remote Client AddOn	*Endgeräte Lizenz je iPhone/iPod Touch (nur in Verbindung mit dem m..Remote Server verwendbar)*

3.2.1 Einspielen einer Lizenzdatei

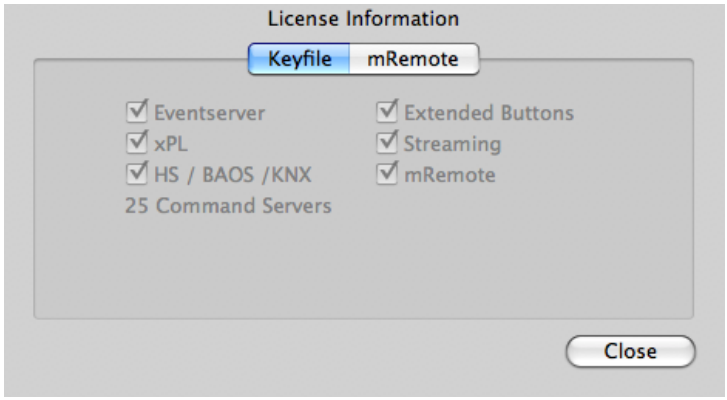
Sollten Sie bereits über eine Lizenz für **m..myhome** verfügen, klicken Sie bitte die Lizenzdatei an und ziehen Sie sie bei gedrückter Maustaste auf das Feld „DROP KEYFILE HERE“:

Lassen Sie die Maustaste los. **m..myhome** wird Ihnen nun mitteilen, ob es sich um eine gültige Lizenzdatei handelt und die Datei auf Ihren Wunsch installieren.

Innerhalb des Feldes für das Keyfile befindet sich ein ein kleiner Button mit einem Fragezeichen:



Wenn dieser Button angeklickt wird, erscheint ein weiteres Fenster, welches Auskunft über die aktuell Installierte Lizenz gibt.



Ist die Lizenz zeitlich limitiert, wird dieses in diesem Fenster ebenfalls angezeigt. Die Funktionen für die mRemote Auswahl befinden sich in der Dokumentation zu den mRemote Produkten.

3.2.2 Anfordern einer Lizenzdatei

Sollten Sie über keine gültige Lizenzdatei verfügen, können Sie **m..myhome** veranlassen, eine entsprechende E-Mail zu generieren. Hierfür muss Ihr auf dem System installierter Email-Client korrekt konfiguriert sein.

Klicken Sie auf „Request Keyfile". **m..myhome** erzeugt daraufhin eine Email mit den zur Lizenzierung notwendigen Systeminformationen und zeigt die erzeuge Email in ihrem Email-Client an. Sollten Sie weitere Anmerkungen zu Ihrer Anfrage haben, können Sie diese selbstverständlich hinzufügen und die E-mail im Anschluss daran absenden. Sie erhalten umgehend Rückmeldung durch unseren Support.

3.3 Steuerung des m..myhome - Dienstes

3.3.1 Aktivieren des Dienstes

Um den Dienst zu starten, Klicken Sie im Bereich „Service Control" auf „Start". **m..myhome** wird daraufhin unverzüglich gestartet. Zusätzlich wird Ihr System so konfiguriert, dass **m..myhome** automatisch gestartet wird, sobald sich der Benutzer, unter dem **m..myhome** installiert wurde, am System anmeldet.

Nach der Installation ist unter „Project Path" ein Verzeichnis für die Projektdaten voreingestellt. Der dort eingegebene und individuell veränderbare Pfad auf das Projektverzeichnis muss immer vollständig angegeben werden. m..myhome erkennt eine Änderung des Ursprungpfades und erzeugt nach Betätigung des „Save & and Restart Buttons" eine entsprechende Verzeichnisstruktur.

3.3.2 Beenden des Dienstes

Klicken Sie im Bereich „Service Control" auf „Stop". Der **m..myhome** -Dienst wird unverzüglich beendet und der Systemeintrag zum automatischen Start (siehe 3.3.1) entfernt.

3.3.3 Statusanzeige

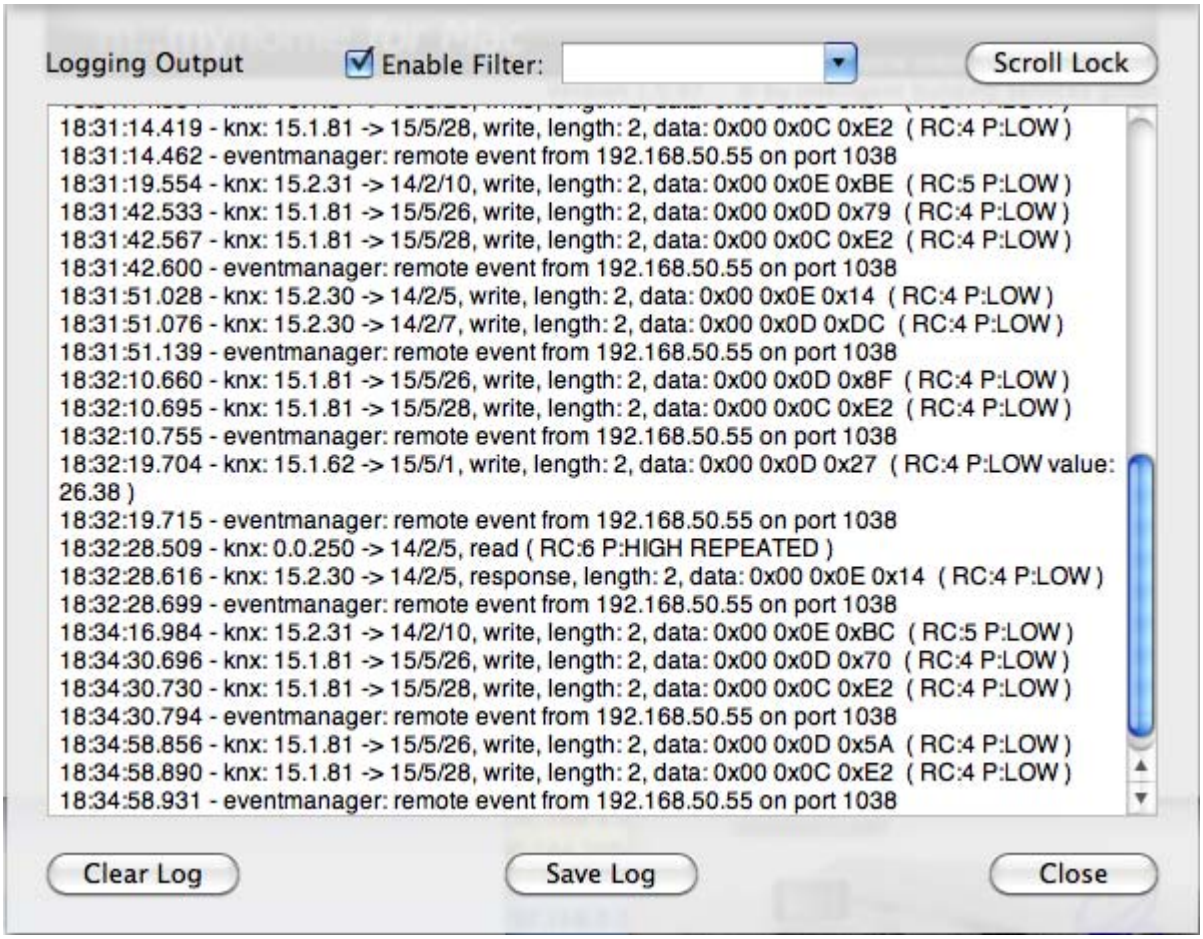
Der jeweils blau hinterlegte Teil im Bereich „Service Control" zeigt den aktuellen Status von **m..myhome** an.

3.3.4 Logging Monitor

m..myhome verfügt über einen Logging Monitor, der Auskunft über die aktuellen Zustände der jeweiligen Services gibt. Ebenso werden Fehler in den Logging Monitor geposted.

Den Logging Monitor erreichen Sie durch Betätigen des Buttons „Show Log" auf der **m..myhome** Konfigurationsseite. Es öffnet sich der Logging Dialog. Die Ausgaben entsprechen den Ausgaben auf Port 1039.

Somit sind zB fehlerhafte Einträge in den Konfigurationsdateien schnell auffindbar. Der Logging Monitor kann auch aufzeichnen und muss für diesen Zweck geöffnet bleiben. Ebenso sind Filtermechanismen eingebaut, die eine gezielte Suche erheblich vereinfachen. Über ein Pulldown Menu können vordefinierte Filter eingestellt werde. Der Eintrag kann jedoch manuell überschreiben werden. Somit sind beliebiger Filteroptionen einstellbar.


Logging Output
Enable Filter:
Scroll Lock
18:31:14.419 - knx: 15.1.81 -> 15/5/28, write, length: 2, data: 0x00 0x0C 0xE2 (RC:4 P:LOW)
18:31:14.462 - eventmanager: remote event from 192.168.50.55 on port 1038
18:31:19.554 - knx: 15.2.31 -> 14/2/10, write, length: 2, data: 0x00 0x0E 0xBE (RC:5 P:LOW)
18:31:42.533 - knx: 15.1.81 -> 15/5/26, write, length: 2, data: 0x00 0x0D 0x79 (RC:4 P:LOW)
18:31:42.567 - knx: 15.1.81 -> 15/5/28, write, length: 2, data: 0x00 0x0C 0xE2 (RC:4 P:LOW)
18:31:42.600 - eventmanager: remote event from 192.168.50.55 on port 1038
18:31:51.028 - knx: 15.2.30 -> 14/2/5, write, length: 2, data: 0x00 0x0E 0x14 (RC:4 P:LOW)
18:31:51.076 - knx: 15.2.30 -> 14/2/7, write, length: 2, data: 0x00 0x0D 0xDC (RC:4 P:LOW)
18:31:51.139 - eventmanager: remote event from 192.168.50.55 on port 1038
18:32:10.660 - knx: 15.1.81 -> 15/5/26, write, length: 2, data: 0x00 0x0D 0x8F (RC:4 P:LOW)
18:32:10.695 - knx: 15.1.81 -> 15/5/28, write, length: 2, data: 0x00 0x0C 0xE2 (RC:4 P:LOW)
18:32:10.755 - eventmanager: remote event from 192.168.50.55 on port 1038
18:32:19.704 - knx: 15.1.62 -> 15/5/1, write, length: 2, data: 0x00 0x0D 0x27 (RC:4 P:LOW value: 26.38)
18:32:19.715 - eventmanager: remote event from 192.168.50.55 on port 1038
18:32:28.509 - knx: 0.0.250 -> 14/2/5, read (RC:6 P:HIGH REPEATED)
18:32:28.616 - knx: 15.2.30 -> 14/2/5, response, length: 2, data: 0x00 0x0E 0x14 (RC:4 P:LOW)
18:32:28.699 - eventmanager: remote event from 192.168.50.55 on port 1038
18:34:16.984 - knx: 15.2.31 -> 14/2/10, write, length: 2, data: 0x00 0x0E 0xBC (RC:5 P:LOW)
18:34:30.696 - knx: 15.1.81 -> 15/5/26, write, length: 2, data: 0x00 0x0D 0x70 (RC:4 P:LOW)
18:34:30.730 - knx: 15.1.81 -> 15/5/28, write, length: 2, data: 0x00 0x0C 0xE2 (RC:4 P:LOW)
18:34:30.794 - eventmanager: remote event from 192.168.50.55 on port 1038
18:34:58.856 - knx: 15.1.81 -> 15/5/26, write, length: 2, data: 0x00 0x0D 0x5A (RC:4 P:LOW)
18:34:58.890 - knx: 15.1.81 -> 15/5/28, write, length: 2, data: 0x00 0x0C 0xE2 (RC:4 P:LOW)
18:34:58.931 - eventmanager: remote event from 192.168.50.55 on port 1038
Clear Log
Save Log
Close

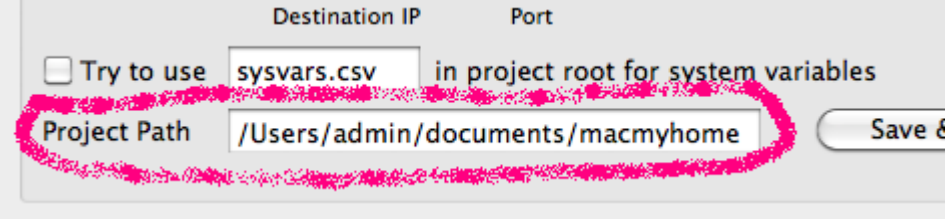
Der „Scroll Lock" Button unterbindet das automatische Scrollen der Einträge. Das Log wird jedoch weitergeschrieben. Der Inhalt des Log's kann in eine Datei ausgelagert werden. Hiefür muss der Button „Save Log" betätigt werden. Darauf hin öffnet sich ein File Dialog, wo der Speicherort und Name des Logfiles eingegeben werden kann. Der Button „Clear LOG" löscht den aktuellen Inhalt des Fensters.

Hinweis: Das Logging funktioniert nur, wenn das PrefPane geöffnet ist, das Logging-Sheet selbst muß nicht geöffnet bleiben. Der Log Puffer beträgt 500KB

3.3.5 Die m..myhome Verzeichnisstruktur

Die m..myhome Verzeichnisstruktur ist mit Ausnahme des Verzeichnisursprungs fest vorgegeben. Wie vor beschrieben, wird diese bei Installation des Services oder bei Änderung des vorgegebenen Pfades automatisch angelegt. Nachfolgend erläutern wir die einzelnen Ordner innerhalb der Projektstruktur.

Der Baum zeigt die Verzeichnisse unterhalb des Verzeichnisursprungs. In diesem Fall ist der Ursprung „macmyhome". Dieser kann, wie bereits erwähnt, beliebig geändert werden.



Die jeweiligen csv Dateien werden nicht automatisch generiert und sollen nur aufzeigen, wo sich die jeweiligen Dateien zu befinden haben. Zusätzlich hinzugefügte Ordner für zB Vorlagedateien sind möglich und beeinflussen in keinster Weise die Funtionalität.

Verzeichnisstruktur

/[macmyhome]/addons - *CommandServer Dateien*

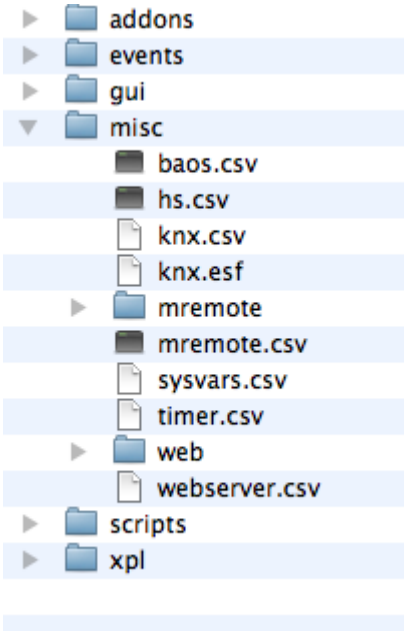
/[macmyhome]/events - *EventServer Dateien*

/[macmyhome]/gui - *Button Leisten Dateien*

/[macmyhome]/misc - *PlugIn's und Systemdateien* für:
- **/baos.csv** - *BAOS Konfiguration*
- **/hs.csv** - *GIRA Homeserver Konfiguration*
- **/knx.csv** - *KNXnet/IP Konfiguration*
- **/knx.esf** - *Die KNX zugehörige OPC- Export-Datei. (Default Name)*
- **/mremote/** - *mremote Projektverzeichnis*
- **/mremote.csv** - *mremote Konfiguration*
- **/sysvars.csv** - *Systemvariabeln*
- **/timer.csv** - *Timer Konfiguration*
- **/web/** - *Bildordner für den Webserver*
- **/webserver.csv** - *Webserver Konfiguration*

/[macmyhome]/scripts *Verzeichnis der Scriptdateien*

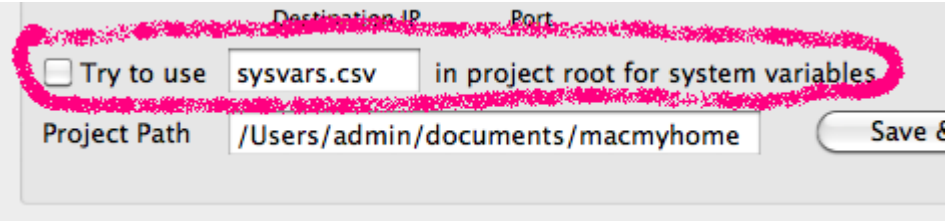
/[macmyhome]/XPL *Verzeichnis der xPL Konfiguration*



Wie die vor genannten Dateien im einzelnen aufgebaut sind, entnehmen sie den nachfolgenden Kapiteln. Das Fehlen einzelner Dateien in der Gesamtstruktur ist unproblematisch. In diesem Fall werden die zugehörigen Schnittstellen deaktiviert. Ebenso ist das Anlegen zusätzlicher Ordner für zB Vorlagedateien unproblematisch.

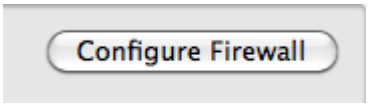
Für komplexe Installationen wurde zus. ein Verweis auf eine Projektweite „sysvar.csv" eingerichtet. Wird

diese Funktion benutzt, wird die Standard **sysvars.csv** in dem „**./misc**" Verzeichnis ignoriert und auf die hier eingestellte Datei verwiesen. Eine genauere Erläuterung dieser Funktion finden Sie in dem Kapitel zum Umgang mit *Systemvariablen*.

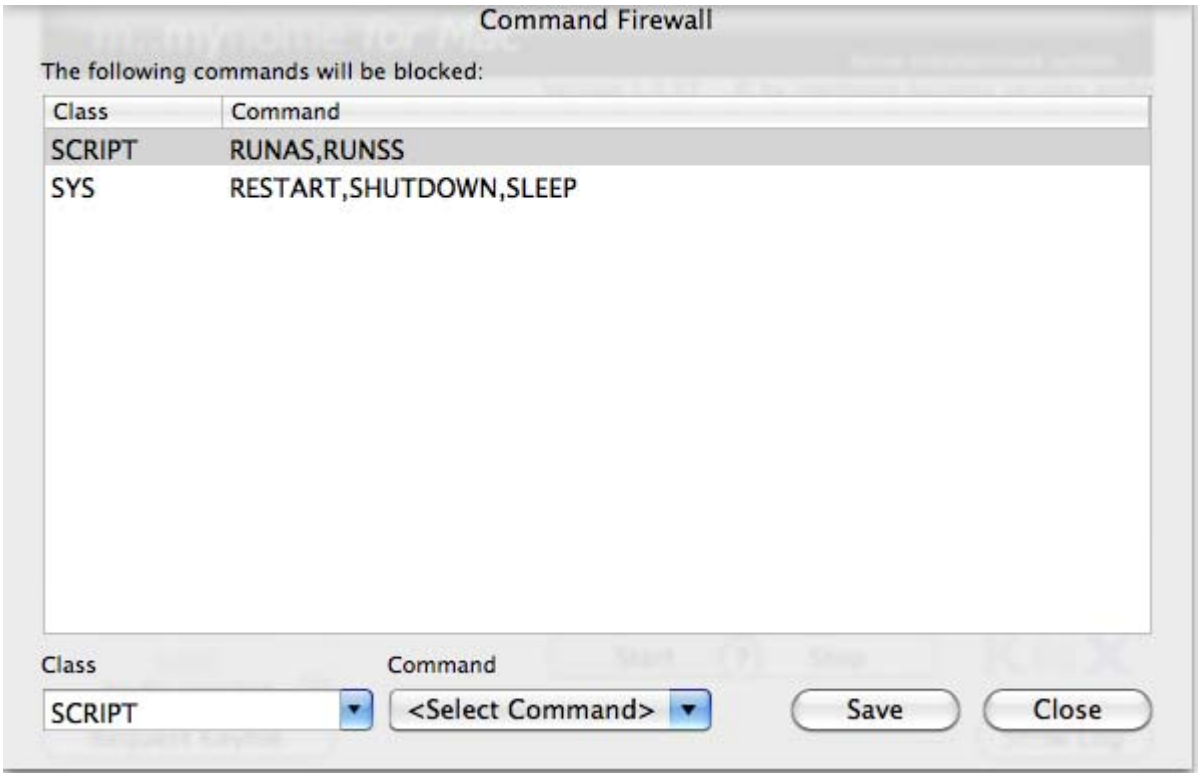


3.3.6 Die m..myhome Firewall

Für zus. Sicherheit und zum Schutz vor Missbrauch ist **m..myhome** mit einer eigenen Firewall ausgerüstet worden.



Diese Firewall schützt lediglich vor der Ausführung von Protokollanweisungen, die auf dem Zielsystem verhindert werden sollen. Man sollte sich bewusst sein, dass bei offengelegten Ports für z.B. den Zugriff über das Internet, auch Code auf das System übertragen werden und per Scriptanweisungen ausgeführt werden könnte. Ebenso könnten Befehle wie z.B. ein **SHUTDOWN** oder ein **SLEEP** für bestimmte Zielsysteme störend sein.



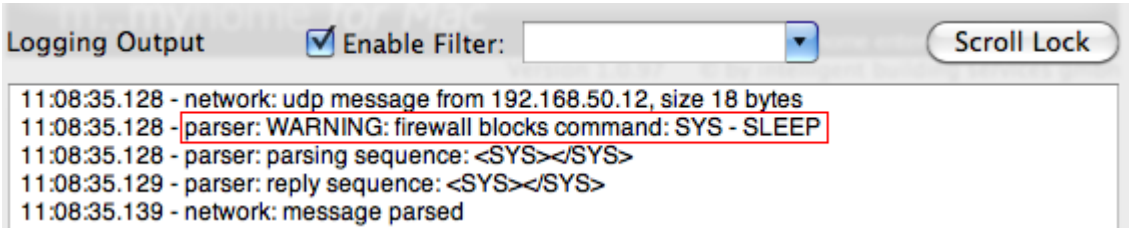
Für die Firewall stehen alle Befehle aller Klassen incl. Der Befehle aus den eingebundenen AddOns

(Command – Server) zur Verfügung. Wertbasierende Befehle werden hierbei nicht berücksichtigt. Die Einrichtung der Firewall ist denkbar einfach. Es stehen für die Klasse und für deren Command's entsprechende Auswahlbuttons zur Verfügung. Hierüber wird der zu ignorierende Befehl einfach ausgewählt.

Ausgewählte Commands sind mit einem Häckchen markiert und können entsprechend auch wieder entfernt werden. Die ausgewählten und zu ignorierenden Befehle/Commands einer Klasse werden in dem Übersichtsfenster zur Anzeige gebracht. Durch die Auswahl von **<ALL>** können alle Befehle einer Klasse markiert werden, die Auswahl **<NONE>** deaktiviert alle Befehle einer Klasse.

Im obigen Beispiel sieht man, dass die Befehle **RUNAS** und **RUNSS** aus der Klasse **SCRIPT** sowie die Befehle **RESTART**, **SHUTDOWN** und **SLEEP** aus der Klasse **SYS** in der Firewall aktiviert und vom **m..myhome** System ignoriert werden. Diese Einstellungen gelten jeweils nur für das Zielsystem, wo die Firewall entsprechend eingerichtet wurde. Bei mehreren m..myhome Systemen im Netzwerk müssen die Einstellungen also je System vorgenommen werden.

Die Firewall Konfiguration wird nur beim Start von **m..myhome** gelesen. Wenn Kommandos/Klassen blockiert werden, wird dies im Log vermerkt.


Logging Output
Enable Filter:
Scroll Lock
11:08:35.128 - network: udp message from 192.168.50.12, size 18 bytes
11:08:35.128 - parser: WARNING: firewall blocks command: SYS - SLEEP
11:08:35.128 - parser: parsing sequence: <SYS></SYS>
11:08:35.129 - parser: reply sequence: <SYS></SYS>
11:08:35.139 - network: message parsed

Wichtige Hinweise:

Die Änderungen werden erst gespeichert, wenn der SAVE Button betätigt wird. Nicht gespeicherte Änderungen gehen erst verloren, wenn das PrefPane geschlossen wird.

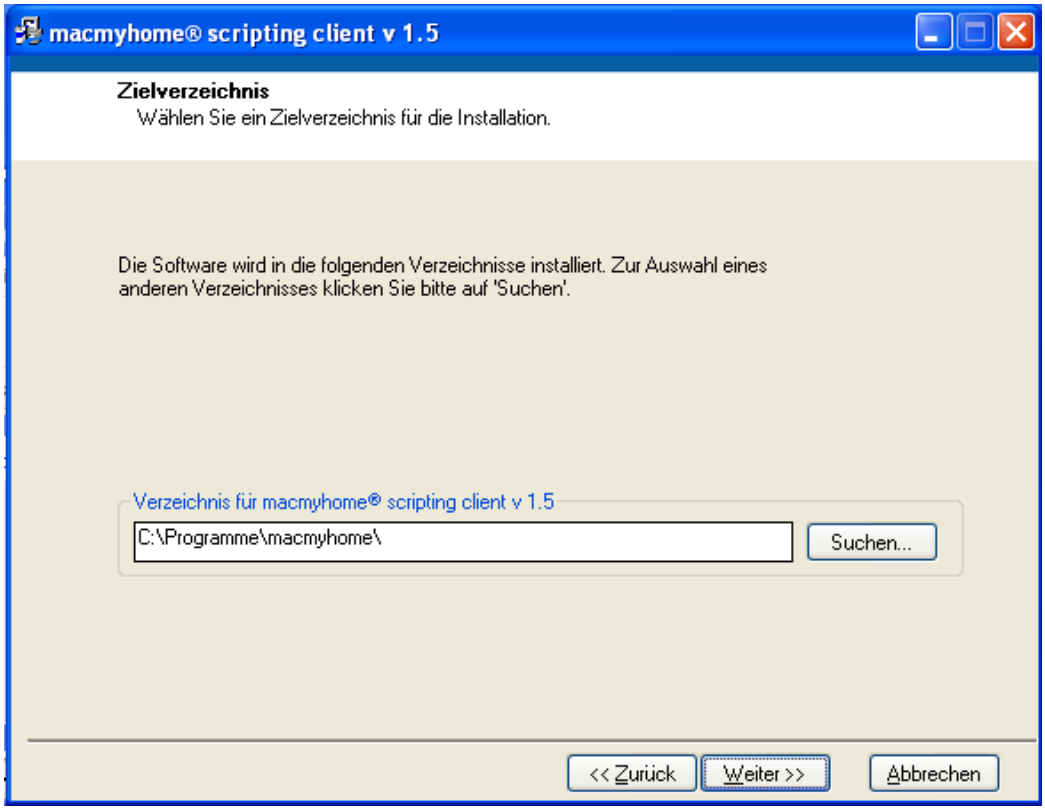
Änderungen an den Firewall Einstellungen werden erst nach einem Service Restart aktiv. CommandServer Klassen werden gelesen, wie dem System bekannt. Werden neue CommandServer Klassen eingebunden, muss auch das PrefPane geschlossen und wieder geöffnet werden, da die Klassen beim Öffnen des PrefPane einmalig gelesen werden.

3.4. Scripting Client Tool

Das Scripting Client Tool ist eine kleine WINDOWS Anwendung, welches bei der Konfiguration des Mac Systems helfen soll. Mit diesem Tool können die Kommandosequenzen für die Steuerung auf einfachem Weg zusammengesetzt werden. Ebenso können die Kommandoketten als Script abgelegt werden. Diverse Logging/Monitoring Funktionen helfen bei der Suche möglicher Probleme. Das Tool wird permanent weiterentwickelt bzw. um weitere Funktionalitäten ergänzt. Diese Dokumentation wird dahingehend nicht permanent ergänzt. Zusätzliche Funktionen sollten in dem Client – Tool selbsterklärend sein.

3.4.1 Installation

Zur Installation der Software öffnen Sie die Datei Setup.exe. Es erscheint das Fenster zur Installation der Anwendung.


macmyhome® scripting client v 1.5
Zielverzeichnis
Wählen Sie ein Zielverzeichnis für die Installation.
Die Software wird in die folgenden Verzeichnisse installiert. Zur Auswahl eines anderen Verzeichnisses klicken Sie bitte auf 'Suchen'.
Verzeichnis für macmyhome® scripting client v 1.5
C:\Programme\macmyhome\
Suchen...
<< Zurück
Weiter >>
Abbrechen

Wählen Sie nun Ihr Zielverzeichnis für die Programminstallation oder übernehmen Sie den vorgeschlagenen Pfad. Bestätigen Sie den Dialog mit „Weiter". Im weiteren Dialog lesen Sie bitte die Lizenzbestimmungen und bestätigen Sie diese.

Nachdem Sie den Dialog mit „Weiter" bestätigt haben, wird die Software installiert.

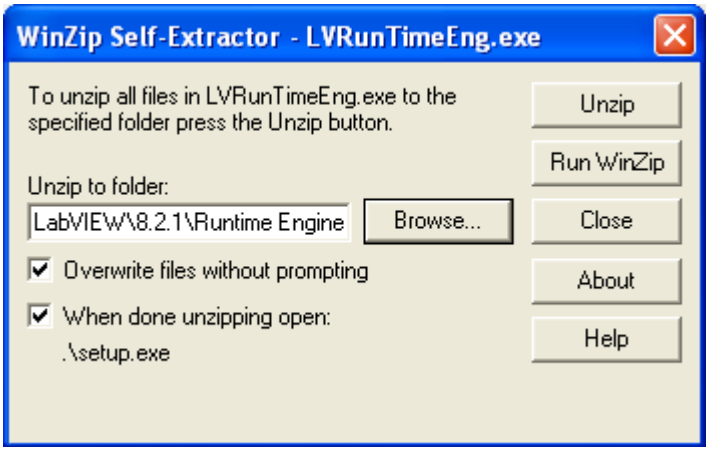
3.4.2 Installation der LabVIEW Runtime Umgebung

Die Software wurde unter LabVIEW entwickelt. Dementsprechend muss die LabVIEW Runtime Umgebung installiert werden. Nutzer die bereits eine LabVIEW Runtime Umgebung auf ihr System installiert haben, können die nachfolgenden Schritt übergehen und die Installationsroutine abbrechen.

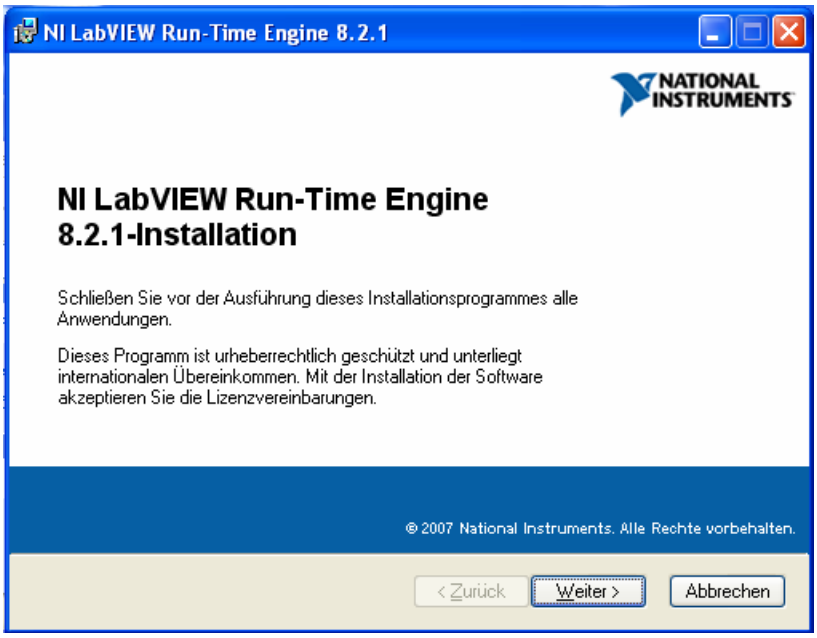
Nach erfolgreicher Installation werden Sie autom. zur Installation der LabVIEW Runtime Umgebung aufgefordert.

MSI
This self-extracting archive will create an installation image on your hard drive and launch the installation.
After installation completes, you may delete the installation image to recover disk space.
You should not delete the installation image if you wish to be able to modify or repair the installation in the future.
OK

Bestätigen Sie das Entpacken der Archivdatei.



Bestätigen Sie den Dialog mit „Unzip". Die Installationsdateien werden nun in ein Temporäres Verzeichnis entpackt und nach Bestätigung des nachfolgenden Dialoges wird die Installation gestartet.


NI LabVIEW Run-Time Engine 8.2.1
NATIONAL INSTRUMENTS
NI LabVIEW Run-Time Engine 8.2.1-Installation
Schließen Sie vor der Ausführung dieses Installationsprogrammes alle Anwendungen.
Dieses Programm ist urheberrechtlich geschützt und unterliegt internationalen Übereinkommen. Mit der Installation der Software akzeptieren Sie die Lizenzvereinbarungen.
© 2007 National Instruments. Alle Rechte vorbehalten.
< Zurück
Weiter >
Abbrechen

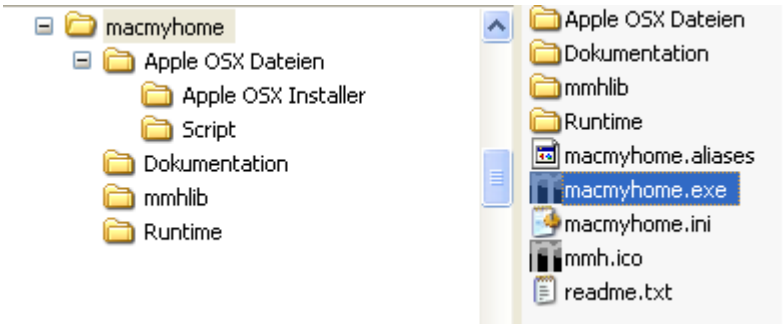
Bestätigen Sie auch diesen Dialog mit „weiter" und akzeptieren Sie im nachfolgenden Dialog die Lizenzvereinbarungen. Bestätigen Sie alle nachfolgenden Dialoge mit „weiter".

Nach erfolgreicher Installation, es erscheint entsprechender Hinweis, müssen Sie Ihr System neu starten. Nach erfolgtem Neustart ist der **m..myhome** Scripting Client betriebsbereit.

3.4.3 Anwendung

Der **m..myhome** Scripting Client dient der einfachen Erstellung von Befehlssequenzen (Script´s), die über dieses Tool generiert und getestet werden können. Ferner können die erstellten Script´s in einer internen Datenbank abgelegt werden. Das Hochladen der Scriptfiles auf Ihren Mac kann ebenfalls über das Tool erfolgen. Ebenso ist eine einfache Schnittstelle eingebunden, die ein einfaches Implementieren neuer Befehlssätze für spätere Versionen ermöglicht.

Als Kontrollmechanismus sind Monitoring Funktionalitäten in dem **m..myhome** Scripting Client eingebunden worden. Dies ermöglicht eine Überwachung der gesamten Kommunikation. Detaillierte Informationen entnehmen Sie dem Kapitel Monitoring.



Nach der Installation des **m..myhome** Scripting Client wurden folgende Dateien auf Ihrem PC System abgelegt. Sollten Sie bei der Installation den Pfad für die Programmdateien nicht verändert haben, befindet sich der Ordner unter c:\Programme.
Neben den **m..myhome** Scripting Client Programmdateien wurden zusätzliche Dateien installiert. In dem Ordner Apple OSX Dateien, Unterordner Apple OSX Installer, befindet sich die Installationssoftware für den **m..myhome** Service für Ihren Mac. Hinweise zur Installation finden Sie in der Datei Hinweis.txt.

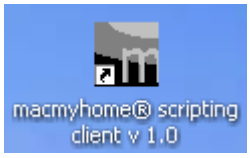
In dem Unterordner Script befinden sich einige Demo Script – Files, die Sie auf Ihren Mac in das Verzeichnis \Dokumente\m. myhome\scripts kopieren können. Sofern das Verzeichnis auf dem Mac nicht vorhanden ist, können Sie es entweder manuell anlegen, oder durch übertragen eines Script – Files aus dem **m..myhome** Scripting Client heraus automatisch erzeugen. Bitte beachten Sie hierbei, dass evtl. vorhandene, gleichnamige Script – Files überschrieben werden. Beachten Sie bitte auch, dass diese Dateien unter dem **m..myhome** Profil abgelegt werden, also dem Profil, wo der **m..myhome** Service installiert wurde. Sofern Sie nicht über ausreichende Kenntnisse für diese Vorgehensweise verfügen, kann das Übertragen der Script – Files ebenfalls über den **m..myhome** Scripting Client erfolgen

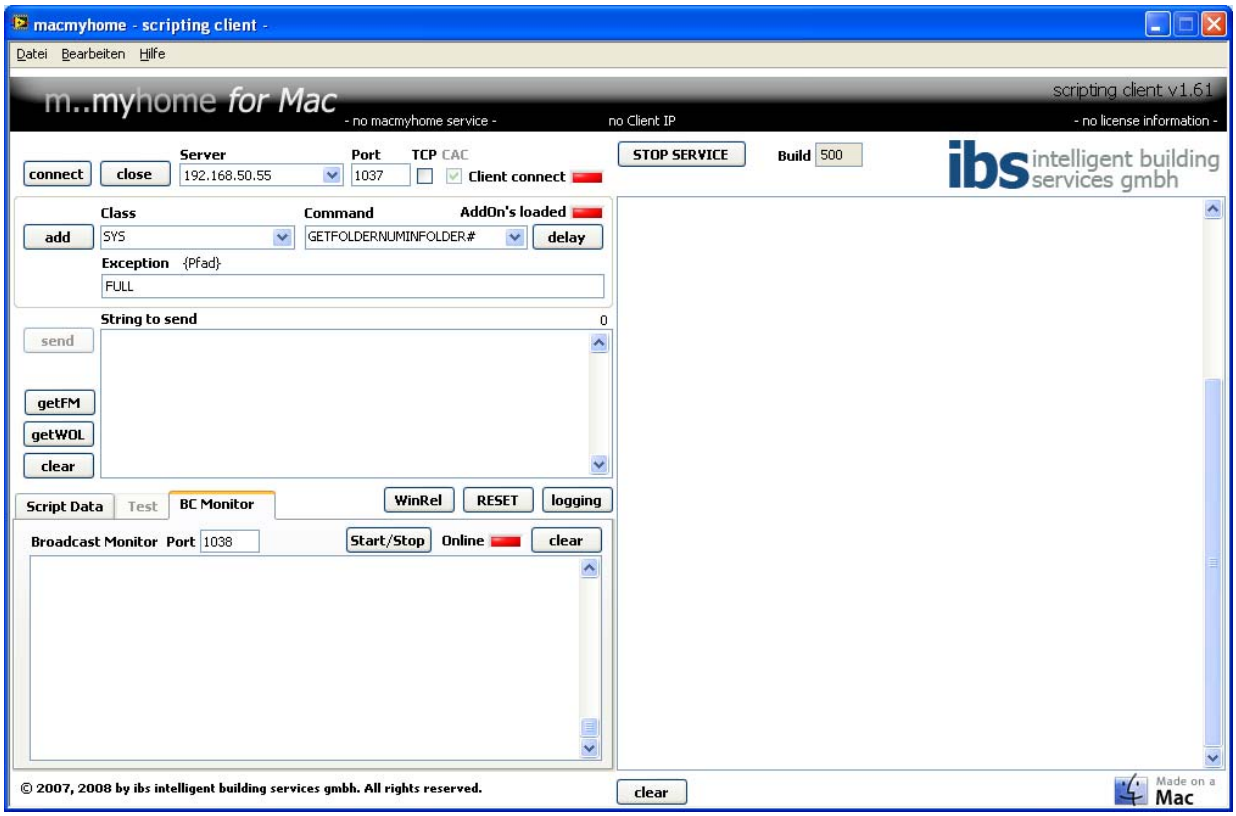
In dem Ordner Dokumentation finden Sie neben diesem Dokument und den Lizen z.B. Bedingungen weitere hilfreiche Dokumente.

In dem Ordner mmhlib befinden sich neben den Programmspezifischen Dateien auch die Command Set Datei CommandList.csv, die den Befehlssatz des **m..myhome** Scripting Client enthält. Diese Datei dient jedoch nur als Backup und kann an einem beliebigen Ort abgelegt werden. Das Laden neuer Befehlssätze erfolgt über einen per Dialog geführten Abruf in dem **m..myhome** Scripting Client. Die Vorgehensweise zum Einlesen neuer Befehlssätze wird im weiteren Verlauf des Dokumentes detaillierter beschrieben.

In dem Ordner Runtime befinden sich, wie vor bereits beschreiben, die Installationsdateien für die LabVIEW Runtime Engine, die für die Nutzung des **m..myhome** Scripting Client zwingend notwendig ist. Anwender die beispielsweise eine B-CON.net Version installiert haben, brauchen die LabVIEW Runtime Engine nicht zusätzlich installieren.

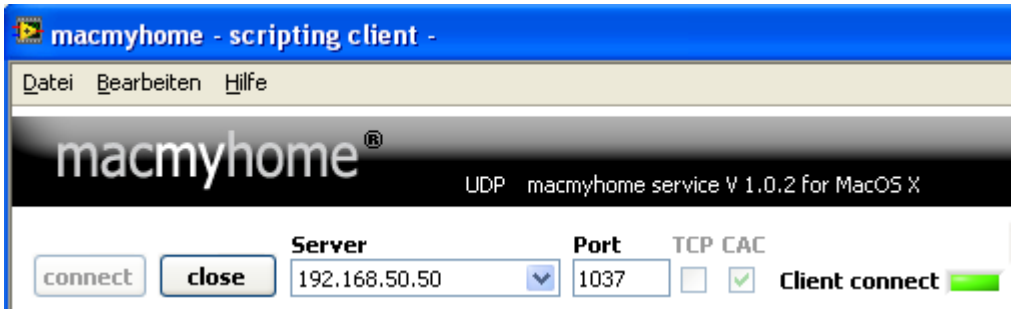
Den **m..myhome** Scripting Client starten Sie durch einen Doppelklick auf das Programm Icon, welches sich auf dem Desktop befindet. Es erscheint nachfolgendes Programmfenster. (Es befindet sich auch ein entsprechender Eintrag im Startmenü)



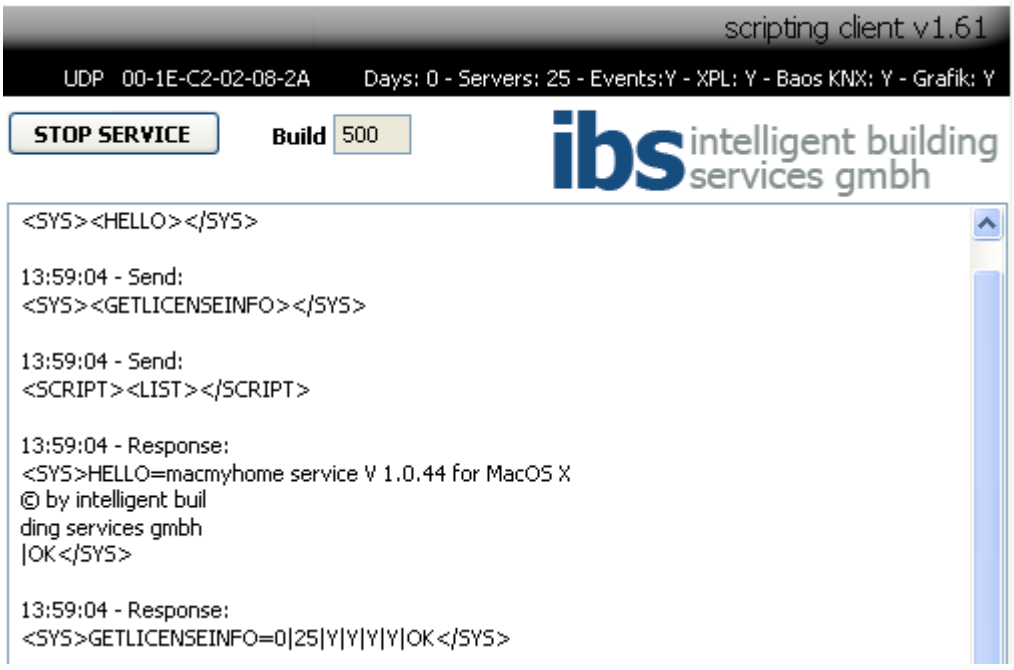


Sofern Sie alle vorherigen Schritte erfolgreich ausgeführt haben, können Sie nun eine Verbindung mit Ihrem Mac System aufbauen. Geben Sie hierfür die IP Adresse und den entsprechenden Port (Standardeinstellung ist 1037) ein. Sie können zwischen einer TCP oder UDP (Standard) Verbindung wählen. Beachten Sie hierbei, dass Sie die Einstellungen konform mit den Einstellungen auf Ihrem Mac System vornehmen.

Wenn Sie alle Eingaben vorgenommen haben, betätigen Sie den Button connect. Die Software merkt sich alle IP Adressen, mit der eine erfolgreiche Verbindung aufgebaut werden konnte. Wollen Sie einen Eintrag aus der Liste entfernen, so wählen Sie den gewünschten Eintrag aus der Liste, positionieren Sie den Mauszeiger über den IP -Adresseintrag und drücken Sie die rechte Maustaste. Es erscheint ein Kontextmenu mit dem Inhalt „Adresse entfernen". Markieren Sie den Eintrag im Kontextmenu mit dem Mauszeiger und betätigen Sie zur Bestätigung die linke Maustaste.



Ist die Verbindung aufgebaut wird dies durch eine Statusanzeige bestätigt. Ebenso wird die aktuell installierte **m..myhome** Version des verbundenen Mac Systems sowie deren MAC Adresse angezeigt. Im Monitoring Fenster können Sie den Datenaustausch beobachten.



Nach dem Kommunikationsaufbau fordert der m..myhome Scripting Client durch die Befehlsketten **<SYS><HELLO></SYS>** die aktuelle m..myhome Programmversion des Verbundenen Mac Systems an und durch **<SCRIPT><LIST></SCRIPT>** die Liste der auf dem verbundenen Mac System gespeicherten Script – Files. Näheres hierzu wird im nachfolgenden Verlauf des Dokumentes erklärt.

Ebenso werden Informationen über die installierte Lizenz sowie die MAC Adresse des verbundenen Apple Systems angezeigt

Hinweis:

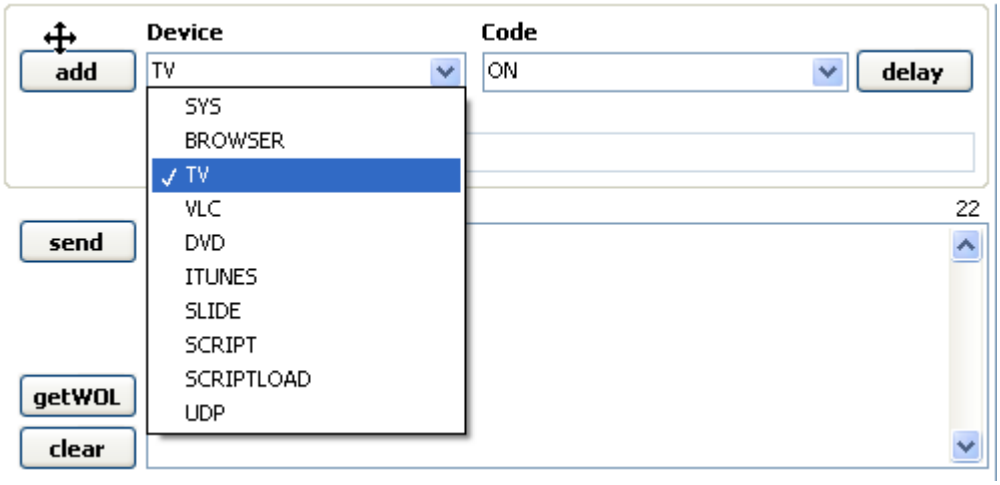
Der Scripting Client aktiviert sich durch eine bestätigte Verbindung mit dem* m..myhome *Service. Ohne bestätigte Verbindung kann der Scripting Client keine TCP/UDP Pakete versenden.

3.4.4 Umgang mit dem Scripting Client

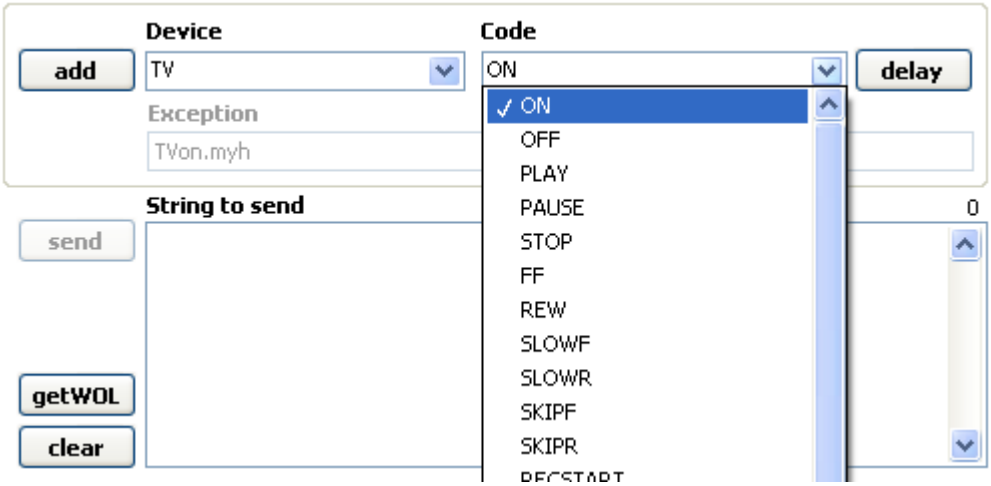
Nachdem Sie nun eine erfolgreiche Verbindung zu Ihrem MAC aufbauen konnten, können Sie beginnen, Befehle an den MAC zu senden. Gehen Sie hierbei genau so vor, wie Sie Ihre Anwendungen lokal an dem MAC bedienen würden. Im nachfolgenden Beispiel wird das Steuern der eyeTV Anwendung erklärt. Nähere Beschreibung zu den möglichen Befehlen sowie deren Funktionen entnehmen Sie dem Kapitel 4 (Befehlssatz).

Der **m..myhome** Befehlssatz gliedert sich in Klassen (Device), dem Klassenbefehl (Code) und einen evtl. zu übergebenden Wert (Exception). Die Codespezifischen Eigenschaften bzw. der notwendige Aufbau der zu sendenden Sequenz wir Ihnen durch den Scripting Client komplett abgenommen. Sie wählen lediglich die Befehle, die Sie senden möchten, über die Auswahlfelder aus und fügen Sie der Sequenz hinzu.

Um nun die Anwendung eyeTV zu öffnen, wählen Sie wie nachfolgend angebildet die Klasse (Device) TV.



Nach Auswahl der Klasse können Sie sich die verfügbaren Klassenbefehle anzeigen lassen und den gewünschten Befehl auswählen. Das Öffnen einer Applikation erfolgt grundsätzlich mit dem „ON" Befehl. Sofern die Applikation bereits geöffnet war, hat der „ON" Befehl keine weitere Auswirkung.



Nachdem Sie nun die Klasse (Device) TV sowie den Befehl (Code) ausgewählt haben, betätigen Sie den Button „add". In der „String to send" Anzeige erscheint nun der vollständige Befehl, der an den **m..myhome** Service gesendet wird.

Betätigen Sie nun den „send" Button um den String an den MAC zu senden. Auf dem MAC startet nun die eyeTV Applikation, sofern diese installiert wurde. Ist Ihr System ohne TV Empfang eingerichtet verwenden eine andere Klasse, z.B. . ITUNES oder DVD.

Nach dem Versand der Befehlskette wird im Monitoring Fenster die erfolgreiche Kommunikation angezeigt. Der **m..myhome** bestätigt (Response) jeden Befehl mit einen „|OK". Kann ein Befehl nicht interpretiert werden, erscheint ein „|NOK".

```
11:37:36 - Send:
<TV><ON></TV>

11:37:36 - Response:
<TV>ON=|OK</TV>
```

Eine erweiterte Logging Funktion ist ebenfalls gegeben und wird im nachfolgendem Verlauf des Dokumentes ausführlicher beschrieben.

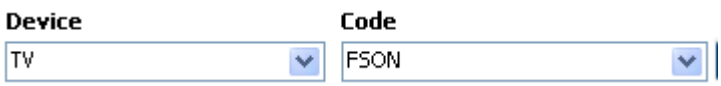
TIP:

Wenn Sie den Mouse Zeiger über den aktuellen Befehl bewegen, erscheint ein Tooltip mit Angabe, welche Funktion der Befehl beinhaltet.

Sie werden feststellen, dass weitere Einstellungen notwendig sind, um die Anwendung in den gewünschten Zustand zu bringen. In der Regel öffnen sich die Anwendungen auf dem MAC nicht immer im Vollbildmodus, wie es für z.B. bei der TV oder DVD Wiedergabe gewünscht ist. Ferner werden weitere Einstellungen benötigt, wie evtl. die Vorgabe der Lautstärke, die Vorgabe der Programmwahl, dass schließen einer vorherig gestarteten Anwendung.

Im nachfolgenden Beispiel werden wir den vorherig erklärten Befehl entsprechend komplettieren, so dass die TV Applikation entsprechend im Vollbildmodus mit definierter Lautstärke und definierter Kanalwahl startet. Bei der Reihenfolge der Befehle gehen Sie bitte ebenso vor, wie Sie es bei der lokalen Bedienung am MAC einstellen würden.

Ausgangsposition ist unser String **<TV><ON></TV>**. Diesen String erweitern wir nun um den Befehl FSON (Vollbildmodus).



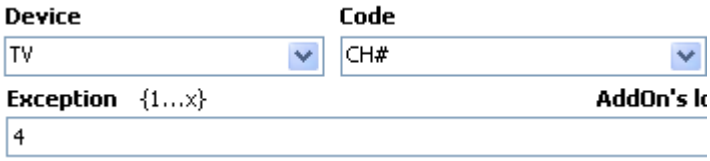
Wählen Sie entsprechend Device und Code aus und betätigen Sie den „add" Button. Beachten Sie bitte, dass die ausgewählten Befehle immer am Ende der Befehlskette angefügt werden. Soll der String an anderer Position eingefügt werden markieren Sie bitte vor dem Betätigen des „add" Buttons die

gewünschte Position im „String to send" mit dem Mauszeiger und bestätigen Sie die Position durch einen Klick auf die linke Mousetaste. Danach betätigen Sie die „add" Taste und der Befehl fügt sich an ausgewählter Position im „String to send" ein. Bitte beachten Sie, dass nun intern die einfüge Position „gemerkt" wird. D.h. alle weiteren Befehle werden immer hinter dem zuletzt eingefügten Befehl angereiht bis Sie die Einfügeposition, wie vor beschrieben, erneut ändern. Sie können ebenfalls händisch diesen String verändern, Einträge oder Wertinhalte löschen bzw. erweitern.

Nachdem Sie nun den „add" Button betätigt haben, sollte der String wie folgt aussehen:

<TV><ON><FSON></TV>

Nun erweitern wir den String um die Kanalvorwahl. Ohne Kanalvorwahl startet eyeTV grundsätzlich mit dem zuletzt eingestellten Kanal. Suchen Sie nun in den Befehlen (Code) der Klasse (Device) TV den Befehl „CH#". Wenn ein Befehl mit dem Sonderzeichen „#" endet bedeutet dies, dass zusätzlich ein Wert erwartet wird, der unter „Exception" eingetragen werden muss. Dementsprechend wird das Eingabefeld aktiviert. Der Wert, der erwartet wird, ist wird hinter „Exception" angezeigt. Für den Befehl „CH" ist dies ein Wert zwischen 1 und der maximalen Anzahl der vorhandenen TV Programme, die eyeTV zur Verfügung stellt.



Tragen Sie nun den Wert Ihrer Wahl in das Feld „Exception" ein und betätigen Sie den „add" Button. Danach sollte der String wie folgt aussehen:

<TV><ON><FSON><CH=4></TV>

Sie können immer wieder mal den String senden um die Auswirkungen auf dem MAC – System zu verfolgen. Sie werden so auch feststellen, dass „doppelt" gesendete Befehlsketten keine störenden Auswirkungen auf das laufende System haben.

Sollte bei Ihnen die Applikation „eyeTV" vor dem versenden des Strings geschlossen gewesen sein, wird das erstmalige Senden dieser Befehlskette lediglich die Applikation öffnen. Dies liegt daran, dass die Applikation erst empfangsbereit ist, wenn sie vollständig gestartet und entsprechend bereit ist. Dies dauert i.d.R. wenige Sekunden. Jedoch hat der **m..myhome** Service die anhängigen Befehle längst an die Applikation gesendet. Da die Applikation aber noch nicht „bereit" war, liefen die Befehle ins Leere. Sie haben hier die Möglichkeit, die Ausführung der Befehle gezielt zu verzögern. Dies wird nachfolgend noch genauer erklärt.

Im nächsten Schritt erweitern Sie nun den Befehlssatz um die Systemlautstärke, mit der die Anwendung starten soll. Wählen Sie hierfür nun in der Klasse SYS den Befehl VOLSET aus. Dieser Befehl erwartet wieder einen zusätzlichen Wert (Exception). Tragen Sie in das Feld einen beliebigen Wert zwischen 0 und 100 ein. Nach dem Betätigen der „add" Taste sollte der String wie folgt aussehen:

<TV><ON><FSON><CH=3></TV><SYS><VOLSET=30></SYS>

Es ist völlig egal, an welcher Steller die Zeichenkette <SYS><VOLSET=30></SYS> nun erscheint, da es keine Auswirkung auf das Ergebnis hat, ob Sie vor dem Öffnen der Applikation oder danach eine Änderung der Lautstärke vornehmen.

Man kann nun noch den String vervollständigen in dem man z.B. Eventualitäten wie einen aktivierten Bildschirmschoner, oder ein vorheriges „Muten" der eyeTV Applikation berücksichtigt und durch entsprechend aufhebende Befehle ergänzt.

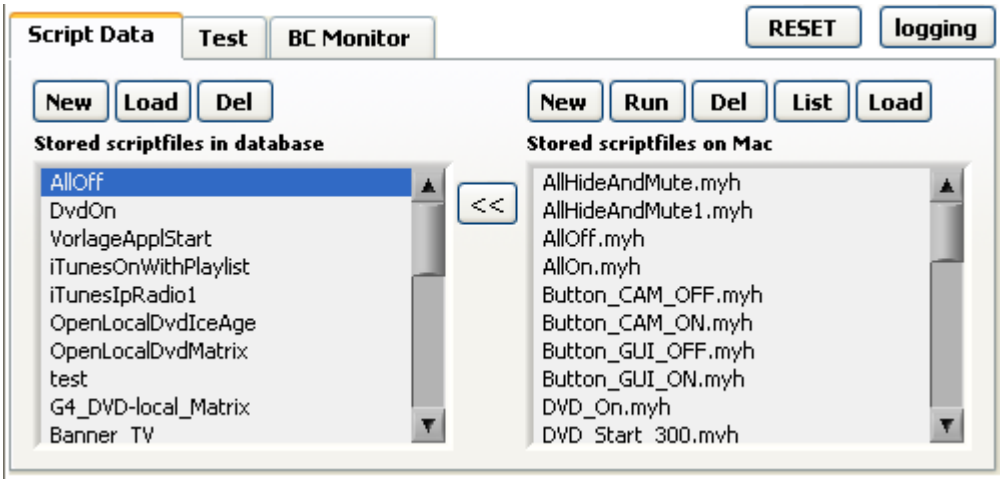
Wählen sie hierfür in der Klasse System DISABLESS und betätigen Sie den „add" Button und wählen Sie im Anschluss aus der Klasse TV den Befehl MUTEOFF und betätigen Sie erneut den „add" Button. Der String sollte nun wie folgt aussehen:

<TV><ON><FSON><CH=3></TV><SYS><VOLSET=30><DISABLESS></SYS><TV><MUTEOFF></TV>

Wenn Sie diesen Befehl nun an den MAC senden, wird die Applikation entsprechend im Vollbildmodus mit der Kanalvorwahl 3, einer Systemlautstärke von 30% starten. Ein evtl. aktivierter Bildschirmschoner wird deaktiviert ebenso wie ein evtl. vorheriges Applikation Mute.

Nun können Sie die gesamte Befehlskette als Script auf dem MAC hinterlegen bzw in die interne Script Datenbank ablegen. Gehen Sie hierbei wie folgt vor:

Wählen Sie in der Registerauswahl des Scripting Client die Karte „Script Data"



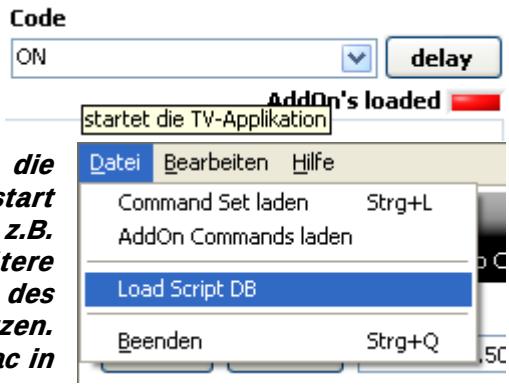
Im linken Teil der Karte befinden sich die Script – Files der internen Script Datenbank und im rechten Teil der Karte befinden sich die ausführbaren Scripts wie sie auf dem Mac abgelegt wurden. Wenn Sie nun den String als Script speichern möchten, betätigen Sie den Button „New" und geben Sie einen Namen für das Scriptfile ein (Bspw. TVon). Die Dateiendung (Suffix) wird automatisch behandelt und ist mit **.myh** fest vorgegeben.

Ist das Scriptfile auf dem Mac gespeichert, wird es in der rechten Auswahltabelle entsprechend angezeigt. Scripts können jederzeit zur Bearbeitung in das „String to send" Fenster geladen werden. Markieren sie hierzu das entsprechende Scriptfile durch einen Mausklick und betätigen Sie den „Load" Button. Möchten Sie das ausgewählte Script löschen, betätigen Sie den „Del" Button und möchten Sie es ausführen (hierbei muss das Script auf dem Mac vorhanden sein), betätigen Sie den „Run" Button.

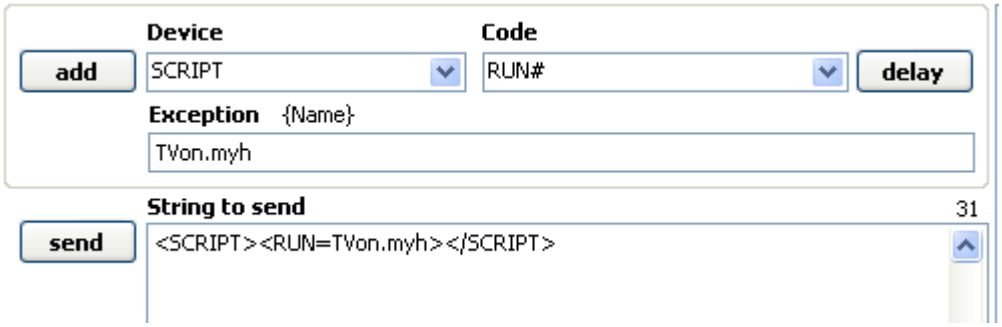
TIP:

Grundsätzlich sollten die Scripts auch immer in der internen Script DB gespeichert werden.

Sie können mehrere interne Datenbanken anlegen bzw. auswählen. Die Datenbank ScriptDB.sdb ist die default Datenbank, die beim Programmstart standardmäßig geöffnet wird. Es empfiehlt sich z.B. projektbezogene Datenbanken anzulegen. Spätere Versionen werden ein direktes kopieren des Datenbankinhaltes auf dem MAC System unterstützen. Ebenso ein direktes kopieren der Script Files vom Mac in die ausgewählte Datenbank.



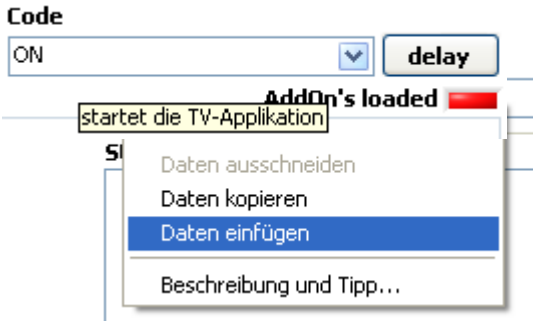
Scripte können jedoch auch über die Klasse „Script" gestartet werden. Wählen Sie entsprechend die Klasse wie nachfolgend dargestellt aus:



Bitte Achten Sie stets darauf, dass bei dem Dateinamen Gross- und Kleinschreibung sowie das Suffix berücksichtigt werden muss.

TIP:

Sie können den Namen des markierten Scriptfiles über die Zwischenablage übernehmen indem Sie durch klicken mit der rechten Maustaste das Kontextmenu öffnen und „copy to clipboard" bestätigen.



Unter „Exception" öffnen Sie dann ebenfalls durch einen Rechtsklick das Kontextmenu und bestätigen „Daten einfügen". Die Daten der Zwischenablage können ebenso in anderen Dialogen verwendet werden.

Mit Ihrem ersten erstellten Script haben Sie nun eine Funktion, die wie vor beschrieben, Ihre TV Anwendung in einem definierten Zustand bringt. Nun möchten Sie aber nicht für jeden einzelnen Programmplatz ein Scriptfile erstellen. Nachfolgendes Beispiel zeigt Ihnen, wie Sie dynamische Argumente (z.B. den Programmplatz/Kanal) dem Scriptfile beim Aufruf übergeben können.

Laden Sie hierzu das Scriptfile wieder in das „Script to send" Fenster (TVon.myh markieren und „Load" Button betätigen). Ändern Sie den String wie folgt ab:

<TV><ON><FSON><CH=[ARG]></TV><SYS><VOLSET=30></SYS>

Speichern Sie das Script erneut ab. Verwenden Sie hierbei einen schlüssigen Namen, wie z.B. „TVon(CH).myh". Wir verwenden diese Art der Beschreibung um zu erkennen, dass das Scriptfile ein Argument, nämlich CH=Kanalnummer, benötigt. Generieren Sie nun den Aufruf Code wie folgt:

<SCRIPT><RUN=TVon(CH).myh><ARG=5></SCRIPT>

Senden Sie den String nun zu dem Mac und Sie sehen, dass die TV Anwendung mit dem gewünschten bzw. vorgegebenen Wert öffnet. Sie können beliebig viele Argumente übergeben. Möchten Sie nun auch die Lautstärke vorgeben ist folgende Änderung notwendig. Ändern Sie erneut Ihr Scriptfile wie folgt ab:

<TV><ON><FSON><CH=[ARG]></TV><SYS><VOLSET=[ARG]></SYS>

Speichern Sie das Scriptfile z.B. unter den Namen „TVon(CH,VOL).myh" und generieren den Aufruf Code:

<SCRIPT><RUN=TVon(CH,VOL).myh><ARG=3><ARG=50></SCRIPT>

Ändern Sie die Argumente nach belieben und senden Sie den String erneut zu dem Mac. Sie werden sehen, dass neben den Programmplatz nun sich auch die Lautstärke einstellen lässt.

Sie können Argumente beliebig verwenden. Argumente können auch anstelle von URL's, Bildpfade, Musiktitel, Fenstergrößen oder auch Code oder Klassen verwendet werden. Das Argument ersetzt oder Ergänzt eine Kommandozeile um den jeweiligen Wertinhalt bevor der Aufruf abgearbeitet wird. Argumente können auch global gefüllt werden. Ein Script mit dem Inhalt:

<[ARG]><ON></[ARG]>

Kann durch Verwendung des Übergabebefehls **GARG** golbal gefüllt werden. Folgender Aufruf würde alle Platzhalter mit dem übergebenen Wert füllen:

<SCRIPT><RUN=Name(Class).myh><GARG=ITUNES></SCRIPT>

ARG= und **GARG=** können auch kombiniert werden. Hierbei ist zu beachten, dass erst **ARG=** zu verwenden ist um gezielt Platzhalter zu füllen. Bei Verwendung eines anschließenden **GARG=** werden alle restlichen Platzhalter gefüllt:.

<TV><ON><FSON><CH=[ARG]></TV><SYS><VOLSET=[ARG]></SYS><[ARG]><ON></[ARG]>

Das Script kann mit folgendem Aufruf entsprechend abgehandelt werden:

<SCRIPT><RUN=Name(CH,VOL,ClassOFF).myh><ARG=3><ARG=50><GARG=ITUNES></SCRIPT>

Entsprechend würde das 1. ARG mit „**3**", das 2. **ARG** mit „50" und das 3. und 4. ARG mit „**ITUNES**" gefüllt werden.

Die Abarbeitung der Argumente erfolgt stets sequentiell.

Im Wesentlichen haben Sie das Prinzip des Scripting Client nun kennen gelernt. Es empfiehlt sich die jeweiligen Befehle einmal nach und nach zu testen um die Auswirkungen auf dem zu steuernden MAC – System kennen zu lernen.

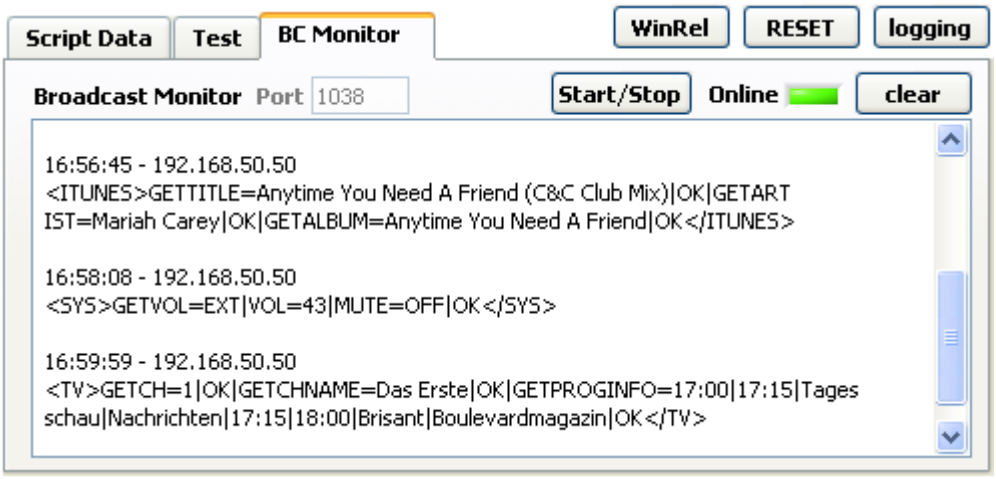
Ein zusätzliches Feature sind die sog. reservierten Scripts. Diese werden automatisch ausgeführt, sofern vorhanden. Mit diesen Scripts können Sie das u.a. das Startverhalten von **m..myhome** beeinflussen.

Reservierte Script Namen:

init.myh	*Der Inhalt des Scripts wird nach einem Systemstart ausgeführt*
exit.myh	*Der Inhalt des Scripts wird nach Beenden des Services ausgeführt*
sleep.myh	*Der Inhalt des Scripts wird vor der Standby Funktion ausgelöst*
wakeup.myh	*Der Inhalt des Scripts wird nach dem „Aufwecken" ausgelöst*

3.4.5 Monitoring

Der **m..myhome** scripting client verfügt neben dem bereits erwähnten Monitoring - Fenster über einen Broadcast-Monitor und einen System-Log-Monitor. Der Broadcast-Monitor dient zur Ansicht der Meldungen, die **m..myhome** über den Broadcast-Port in Netzwerk überträgt. Um den Broadcast-Monitor zu starten, wechseln Sie auf die Karteikarte „BC Monitor".



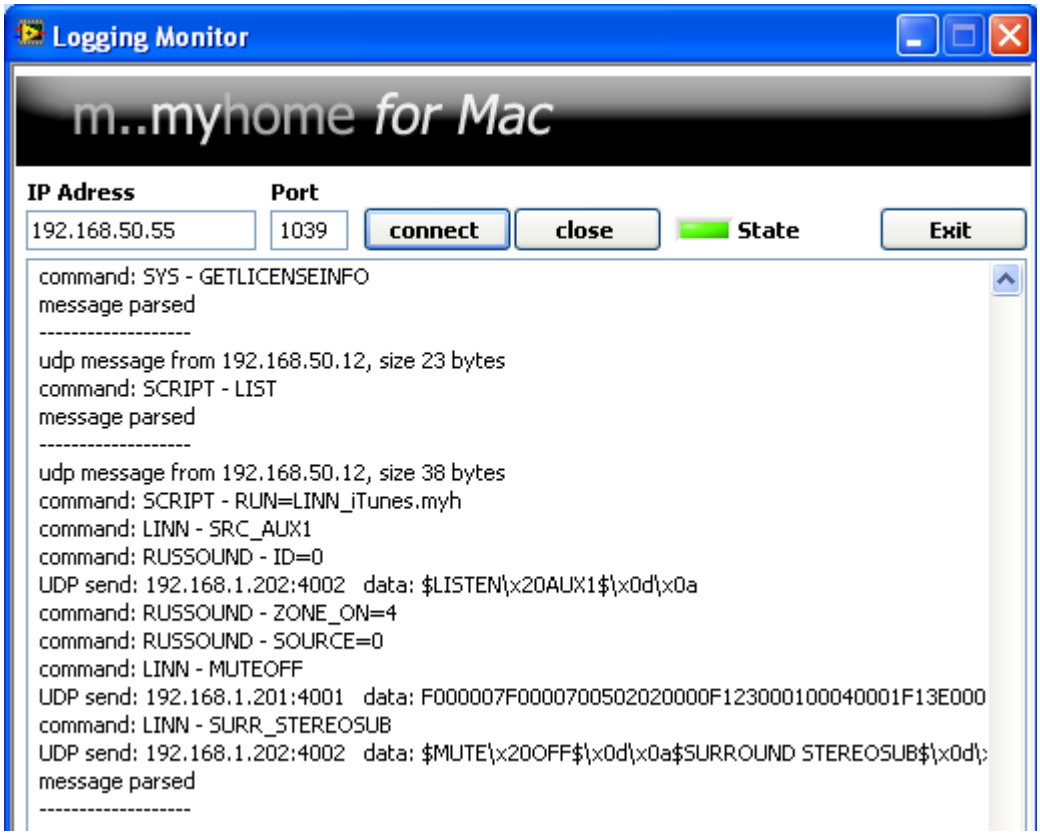
Betätigen Sie den Button „Start" und ändern Sie über Ihre Mac Tastatur z.B. die Lautstärke Ihres Mac Systems. Sie werden sofort eine Feedback Massage mit entsprechend den aktuellen Werten erhalten. Ebenso sendet **m..myhome** Infos zum aktuell laufenden Musiktitel oder dem aktuell laufenden TV Programm. Auch der **PLAYER STATUS** der jeweiligen Anwendungen wird übertragen für iTunes bspw. ein „**PLAYED**", oder „**PAUSED**".

TIP:

Broadcast Status Meldungen können auch über den Eventserver abgefangen werden um wiederum entsprechende Aktionen auszulösen. Der Eventserver wird im weiteren Verlauf des Dokumentation noch genauer beschrieben

---

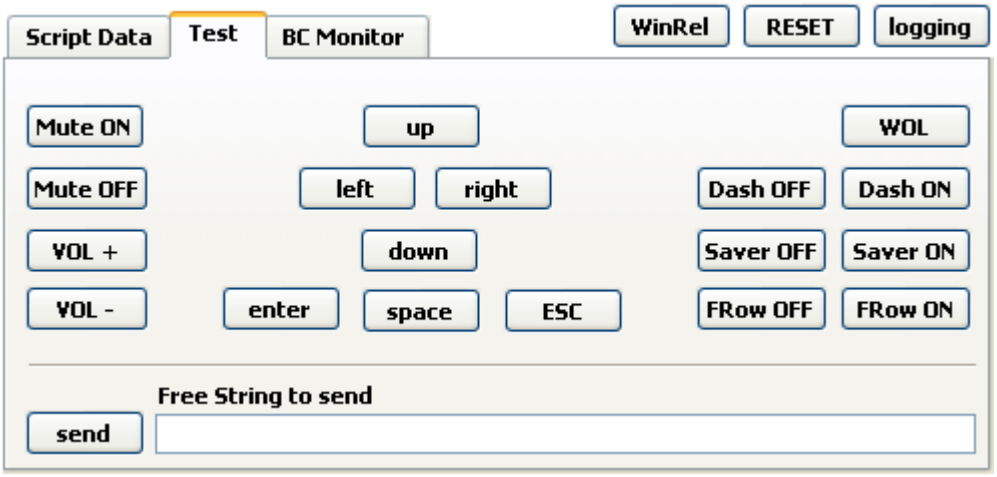
Eine weiteres Monitoring Fenster öffnet sich durch Betätigen der „logging" Taste. Dies ist der Systemmonitor, der Auskunft darüber gibt, was der **m..myhome** Service abarbeitet. Dieser Monitor dient z.B. der Analyse sofern Probleme eintreten bzw. kann unplausibles Verhalten aufzeigen. Ebenso können die Sendesequenzen bei Verwendung der CommandServer Funktion beobachtet werden.



Der Monitor muss durch Betätigung des „connect" Buttons gestartet werden. Der Port 1037 ist fest eingestellt und darf nicht verändert werden.
Ab der Version 1.0.46 wurden die Ausgaben auf dem Loggingport erweitert. Es wird jetzt sowohl die komplette, empfangene Kommandosequenz wie auch die Rückgabesequenz ausgegeben. Außerdem werden sämtliche Commandserver-Replays mit Ziel-IP und -Port ausgegeben.

3.4.6 Testfeld

Um grundsätzliche Steuerfunktionen schnell zu testen haben wir ein kleines Testfeld in den Scripting Client eingebunden. Dies finden Sie unter der Karteikarte „Test".



Die Buttons sind mit spezifischen Systemfunktionen vorbelegt. Sobald der Client mit Ihrem Mac verbunden ist, lassen sich die Funktionen ausführen. In dem Feld „Free String to send" können freie Befehlsketten an den Mac gesendet werden. Die Daten werden an den entsprechend verbundenen Mac gesendet. Die Funktionen der jeweiligen Buttons entnehmen Sie den Hinweisstreifen (Tipstrip/Mouseover)

TIP:

Benutzen Sie den „Free String to send" um z.B. auch häufig verwendete Befehlsketten oder Script Aufrufe zu „parken".

3.4.7 Weitere Funktionen

Im Dateidialog des **m..myhome** Scripting Client finden Sie einige weitere Funktionen. Diese beinhalten neben dem bereits beschriebenen Laden der ScriptDB auch eine Funktion zum aktualisieren des Befehlssatzes (Command Set) sowie den Abruf evtl. vorhandener CommandServer Definitionen (AddOn).

Sobald Ihr System über entsprechende Definitionstabellen verfügt, wird der Befehlssatz bei Abruf der Daten erweitert. Hinweise zu dem CommandServer Dateien finden Sie im weiteren Verlauf der Dokumentation.

Mit der Funktion „Command Set laden" können Sie den aktuellen Befehlssatz aus der Definitionsdatei „CommandList.csv" einlesen. Die aktuellste Definitionsdatei finden Sie im Upload Verzeichnis auf unserer Webseite *www.macmyhome.de* .

4. Der m..myhome Befehlssatz

Nach dem Start der Applikation bindet **m..myhome** auf allen im System verfügbaren Netzwerk-Schnittstellen die in der Konfiguration definierten Ports an sich und wartet auf eingehende Verbindungen. Erhält **m..myhome** ein Kommando, das dem definierten Kommandosatz entspricht und eine gültige Syntax aufweist, führt er dieses Kommando aus und meldet die erfolgreiche Ausführung sowie ggfs. auftretende Rückgabewerte zurück an den aufrufenden Client.

Je nach Art des Kommandos werden hierbei unterschiedlichste Aktionen im System ausgelöst. Der Umfang reicht von einfachen, die Benutzeroberfläche des Systems betreffenden Aktionen bis hin zum kompletten Abschalten bzw. Neustarten des Systems.

m..myhome bietet weiterhin die Möglichkeit, verschiedene Scripte und Befehlssequenzen lokal abzuspeichern und bei Bedarf vollautomatisch auszuführen.

4.1 Begrifflichkeiten / Typen

Der Befehlssatz der **m..myhome** - Software untergliedert sich in zwei Befehlstypen:

4.1.1 Klassen (CLASS)

Klassen (CLASS) dienen dazu, **m..myhome** mitzuteilen, welcher Befehlstyp ausgeführt werden

soll. Außerdem wird so der unterstütze Befehlssatz in einem überschaubaren Rahmen gehalten. Weiterhin müssen Klassenaufrufe immer durch </CLASS> terminiert werden (siehe 4.2), sodass mit diesem Verfahren gleichzeitig eine Sicherung gegen unbeabsichtigt unterbrochene Verbindungen und unvollständige Sequenzen gewährleistet ist. Ein Klassenaufruf führt noch keine Befehle im **m..myhome** aus.

4.1.2 Kommandos (CMD)

Kommandos (CMD) stellen den eigentlichen Befehlssatz des **m..myhome** zur Verfügung. Sie können eigenständig oder mit Argumenten versehen sein.

4.1.3 Scripte

Ein Script ist eine Kette von Kommandos, dass als aufrufbare Funktion zur Verfügung steht. Es können beliebig viele Scripte erzeugt und abgerufen werden. Ferner können dem Script auch beliebige Daten beim Aufruf übergeben werden.

4.1.4 Sonderkommandos

Sonderkommandos sind klassenunabhängig und können an beliebigen Stellen in der Sequenz eingefügt werden. Es gelten die syntaktischen Regeln der Standardkommandos. Es wird jedoch empfohlen, Sonderkommandos außerhalb einer Klasse zu verwenden.

4.1.5 Formatierungsoptionen (optional)

Formatierungsoptionen können **optional** den jeweiligen Kommandos angefügt werden. Über entsprechende Formatstrings können Ausgaben beliebig beeinflusst werden. Ebenfalls können hierüber Such- und Sortierfunktionen eingeleitet werden

4.1.6 Konverter bzw Konvertierungsoptionen (optional)

Konvertierungsoptionen können **optional** verwendet werden um Argumente zu beeinflussen, umzurechnen oder Umzuwandeln

4.2 Syntax (Aufbau des Protokolls)

Die Syntax für einen gültigen Befehl **ohne Argument** lautet:

<CLASS><CMD></CLASS>

Die Befehlssequenz für einen Befehl **mit Argument**:

<CLASS><CMD=ARG></CLASS>

wobei **ARG** für das einzusetzende, kommandoabhängige Argument steht.

Die Befehlssequenz für einen Befehl **mit Formatierungsoptionen**:

<CLASS><CMD{FORMATSTRING}=ARG></CLASS>

Wobei der optionale **{FORMATSTRING}** für den entsprechend einzusetzenden Ausdruck steht.

Die Befehlssequenz für einen Befehl **mit Konvertierungsoptionen**:

<CLASS><CMD=ARG{KONVERTERSTRING}></CLASS>

Wobei der optionale **{KONVERTERSTRING}** für den entsprechend einzusetzenden Ausdruck steht. Formatierungs- und Kovertierungsoptionen können auch gleichzeitig innerhalb eines Kommandos Anwendung finden.

Kommandos lassen sich kaskadieren, so dass eine Befehlssequenz entsteht:

<CLASS><CMD1><CMD2><CMD3=ARG1><CMD4>.....<CMDn></CLASS>

Auf dieselbe Weise lassen sich verschiedene Klassen in einer Sequenz kombinieren:

<CLASS1><CMD1></CLASS1><CLASS2><CMD2></CLASS2><.....></CLASSn>

Auch lassen sich 1Byte Hexwerte in die Kommandosequenz einbinden. Hierüber können zB auch nicht sichtbare Zeichen übergeben werden. Die Übergabewerte werden in die entsprechende dezimale Zahl gewandelt. Hexadezimale Werte werden durch ein vorangestelltes „**\0x**" kenntlich gemacht.

Beispiel:

<SYS><SETBUTTONLABEL=main,2,Zeile 1\0x0dZeile 2></SYS>

Schreibt einen zweizeiligen Text in die Hauptbuttonleiste.

Die maximale Länge einer Befehlssequenz beträgt 8192 Byte.

Wird eine Klasse nicht terminiert, so werden alle in diesem Klassenaufruf enthaltenen Befehle verworfen.

4.2.1 Iterative Abarbeitung

Das m..myhome Protokoll unterstützt ebenfalls Iterative Prozesse. So können zB auch Array's bzw Listen verwendet werden. Ein Array definiert sich durch das Trennzeichen „|". Hierüber ist es zB möglich, Mehrfachabfragen zu generieren. Wird zb einem Kommando als Argument eine solche Liste übergeben, wird intern diese Liste iterativ abgearbeitet und eine entsprechende Antwortsequenz erzeugt. Dies Verhalten ist gleichzusetzen mit einer „FOR-Schleife", wie man sie aus der Informatik kennt.

Beispiel:

<ITUNES><GETIDTITLE=1080></ITUNES>

Würde als Antwortstring beispielsweise folgende Ausgabe generieren:

<ITUNES>GETIDTITLE=One Of The Brightest Stars|OK</ITUNES>

Wird nun aber eine Liste mit mehreren Titeln übergeben wie zB:

<ITUNES><GETIDALBUM=1040|1046|1052|1058|1062|1064|1066></ITUNES>

Erfolgt die Ausgabe entsprechend:

<ITUNES>GETIDTITLE=Carry You Home|Give Me Some Love|I'll Take Everything|One Of The Brightest Stars|Same Mistake|OK</ITUNES>

Inten zerlegt **m..myhome** die übergebene Liste in deren einzelne Elemente und führt entsprechend ein GETIDALBUM mit dem jeweiligen Element aus. Das Ergebnis wird wiederum als Liste ausgegeben.

4.3 Antwortsequenz

Die Antwortsequenz der **m..myhome** - Software besteht aus einer Wiederholung der Kommandosequenz. Argumente werden nicht wiederholt. Zwecks XML-Kompatibilität wird bei den Kommandos auf die umfassenden „<" und „>" verzichtet. Mehrere Kommandos derselben Klasse werden durch „|" getrennt.

Syntaktisch gültige Kommandos werden in der Antwortsequenz immer mit dem Argument „OK" versehen, ungültige Kommandos mit „NOK":

<CLASS1>CMD1=OK|CMD2=NOK</CLASS1><CLASS2>....

Im obigen Beispiel war **CMD1** ein gültiges Kommando; **CMD2** war ungültig.

Enthält die Antwort auf ein Kommando ein Argument (**ARG**), wird das Argument zuerst hinter dem Befehl eingefügt:

<CLASS>CMD=ARG|OK</CLASS>

Ebenso verhält es sich bei der Verwendung von Formatoptionen. Die Parameter werden ebenfalls der Antwortsequenz angefügt.

<CLASS>CMD{FORMATSTRING}=ARG|OK</CLASS>

Die maximale Länge einer Antwortsequenz kann 8129 Byte betragen.

4.4 Formatieroptionen, Parser Optionen, Status Port

Nachfolgend beschreiben wir verwendete Platzhalter für diverse Funktionalitäten, wie dem Einleiten von Abfragen mit formatierter Ausgabe, dem Auswerten bzw. Parsen von Textinformationen sowie der Informationsübergabe.

4.4.1 Formatieroptionen.

Verwendung von Formatoptionen, die formatierte Rückgabewerte erzwingen können. Dies ist zB notwendig um Timeticker, die in Sekunden ausgegeben werden, automatisch in Minuten und Stunden zu formatieren. Ebenso können hierüber gezielte Listenausgaben generiert werden, wie sie zB bei einer iTunes Playlisten Abfrage notwendig sind.

Die Option wird an **dem Befehl**, deren Rückgabewert ausgewertet werden soll, in „{}" eingefasst angehängt.

Folgende Formatierungsoptionen sind z.Z. verfügbar:

%h *- rechnet einen Wert in volle Stunden um*
%m *- rechnet den Wert in volle Minuten um*
%s *- stellt den Wert 1:1 dar*
%l *- (Liste) initiiert eine nach „Index", „Länge" indizierte Ausgabe*
%lc *- (count) Zählt die Elemente einer Liste (sollte am Ende in der Optionskette stehen)*
%ls *- (sort) Sotiert eine Liste alphabetisch*
%lf *- (filter) Filtert die Ausgabe der Liste*
%lu *-(unique) Filtert alle mehrfach vorhandene Elemente aus*
%c *- (Teilstring) gibt einen Teilstring aus*
%r *- (Ersetzen) ersetzt eine Zeichenkette durch eine Neue.*
%t *- Dummy, um z.b Listenausgaben zu bennen (Tagging).*

Formatoptionen lassen sich beliebig kombinieren bzw verketten. Hierbei werden die Formatoptionen durch ein komma getrennt, **{%lfa,%ls ,%lc }** (Erzeugt eine Liste deren Elemente mit „a" beginnen und sortiert die Elemente alphabetisch und stellt die Summe aller Elemente der Liste voran). Beachten Sie hierbei bitte, dass die Optionen der Reihe nach angewandt werden. So sollte natürlich die Filterung erfolgen, bevor der Count ausgegeben wird.

Anwendung der der verschiedenen Optionen:

Erzwingen von Stellen:

%3m - erzwingt mindestens 3 Stellen und stellt blanks voran

Erzwingen und Auffüllen von führenden Nullen:

%04s - erzwingt mindestens 4 Stellen und füllt mit führenden Nullen auf

Gezieltes Ausgeben von Listen (möglich bei allen Befehlen, die Listen zurückgeben)

%l{Index},{Länge} - gibt ab {Index} {Länge} Listenelemente zurück. wenn Länge negativ ist, wird rückwärts gezählt, also die Ausgabe herumgedreht

%lc Fügt der Listenausgabe die Anzahl der beinhalteten Elemente im Format „...=({Anzahl})|.." an

%ls Sortiert die Listenausgabe alphabetisch

%lf{Suchstring} Filtert die Listenausgabe nach dem Inhalt von {Suchstring}. Hierbei erfolgt die Suche von links nach rechts.

%lu Ignoriert Elemente der Liste, die mehrfach ausgegeben werden.

Universelle Format-String

%c{Index},{Länge} - *gibt bestimmte Zeichen eines Strings aus.*

%r{String A},{String B} - *ersetzt alle Zeichen (String A) durch (String B).*

Rückgabewerte formatieren (bisher nur für Standard-Kommandos möglich, nicht für Commandserver Replays).

Beispiele für die Verwendung der Formatoptionen:

Der normale <ITUNES><GETCTIME></ITUNES> Aufruf liefert im Abspielmodus z.B. 521 Sekunden zurück.
Der Aufruf:

<ITUNES><GETCTIME{%02m:%02s}></ITUNES>

liefert:

GETCTIME=08:41|OK

Es werden erst Minuten und Sekunden berechnet. Dann werden die Minuten 2-stellig mit führender Null dargestellt, ein ":" als Trenner eingebaut und anschließend die Sekunden 2-stellig angefügt. Dementsprechend liefert der Aufruf:

<ITUNES><GETCTIME{%02h:%02m:%02s}></ITUNES>

die Ausgabe

GETCTIME=00:08:41|OK

Will man nur führende Nullen für einen Wert erzwingen, trägt man nur den "Sekunden"-Formatierer ein:

<ITUNES><GETCTIME{%07s}></ITUNES>

liefert die Ausgabe

GETCTIME=0000521|OK

Sollen Argumente formatiert werden (funktioniert prinzipiell genau wie oben) ist zu beachten, dass der Formatstring hinten an das Argument angefügt wird:

<SYS><SETVOL=87{%06s}></SYS>

wird intern zu:

<SETVOL=000087>

Beispiele für die Listenausgabe.

<SYS><GETFOLDERINFOLDER{%l17,5}=/Users/mmh/Test></SYS>

Liefert 5 Ordnernamen ab dem Index 17 als Liste zurück

<SYS><GETFILESINFOLDER{%l12,-5}=/Users/mmh/Test></SYS>

Liefert die Dateinamen von Index 12 bis Index 8, beginnend mit Index 12

<ITUNES><GETALLALBUMS{%lc}></ITUNES>

Liefert wie folgt die Namen aller Albums aus der iTunes Datenbank und stellt die Anzahl der Elemente (Treffer) der Ausgabe voran:

<ITUNES>GETALLALBUMS{%lc}=(4)|All The Lost Souls|Café del Mar: The Ibiza Clubmixes (2008)|Embrya|Lounge Top 55 Deluxe|Pacha - The World's Most Famous Club Sound 2008|OK</ITUNES>

<ITUNES><GETIDTITLE{%ls}=1080|1082|1084|1086|1088></ITUNES>

Liefert den Titelnamen zu den abgefragten ID's und gibt die Namen sortiert zurück.

<ITUNES><GETALLALBUMS{%lfc}></ITUNES>

Liefert eine Liste aller Alben, die mit „c" beginnen. (nicht case sensitiv)

<ITUNES><GETALLALBUMS{%lc,%lu}></ITUNES>

Liefert eine Liste aller Alben und filtert doppelt vorhandene Albennamen aus.

Es können Formatoptionen beliebig kombiniert werden. So liefert zB der Aufruf:

<ITUNES><GETALLALBUMS{%lfa,%lc,%ls}></ITUNES>

eine Liste aller Alben beginnend mit dem Buschstabe „a", stellt die Anzahl der Treffer dem Ausgabestring voran und sortiert die Liste alphabetisch.

Beispiele für universelle Formatstrings.

<ITUNES><GETTITLE></ITUNES>

Liefert beispielsweise **„Cafe Del Mar"**

<ITUNES><GETTITLE{%c5,3}></ITUNES>

Gibt den String ab Position 5 in der Länge 3 aus, also **„Del"**

<ITUNES><GETTITLE{%c5,14}></ITUNES>

Liefert den String **„Del Mar * Cafe"**. Gibt den String ab Position 5 in der Länge 14 aus. Da die gewünschte Länge des Strings größer ist als der verbleibende Rest des Strings, wird " * " eingefügt und vom Beginn des Strings erneut gelesen, bis die 14 Zeichen erreicht sind.

Wenn der {Index}-Wert größer als die Stringlänge ist, wird er automatisch heruntergerechnet. Ein "Hinauslaufen" bei hohen {Index}-Werten ist also nicht möglich

<SQUEEZEBOX><ADDANDPLAY=Die besten Hits{%r,%20}></SQUEEZEBOX>

wandelt den Befehl in <ADDANDPLAY=Die%20besten%20Hits> um, ersetzt also also alle „Blanks" in %20.

<ITUNES><GETIDTITLE{%tmRemotePlaylist}=518|520|522></ITUNES>

Erzeugt die Antwort:

<ITUNES><GETIDTITLE{%tmRemotePlaylist}=Hello|Bad|Erotica|OK</ITUNES>

Wenn man nun auf **GETIDTITLE{%tmRemotePlaylist}=*|OK** matcht, kann man die Antwort eindeutig zuordnen ohne Gefahr zu laufen, daß eine andere Abfrage zur Erzeugung einer neuen Liste herangezogen wird. Beispielsweise können hierüber auch mögliche Matchbedingungen in Event Definitionen allein auf den Tagnamen formuliert werden, also zB auf **{%tmRemotePlaylist}=*|OK.** Der Initiator (**GETIDTITLE** oder **GETIDALBUM**) der Liste bleibt hierbei völlig unberücksichtigt. So können zB Anzeigen(m..Remote) mehrfach verwendet werden.

%t hat also keine echte Funktion, sondern bietet lediglich ein Unterscheidungsmerkmal.

4.4.2 Konvertieroptionen.

Verwendung von Konvertierungsoptionen, die ein Argument umrechnen/umwandeln oder auch konvertieren können. Dies ist zB notwendig um Werte in HEX zu wandeln oder Werte gezielt zu berechnen, wie nachfolgende Beispiele verdeutlichen sollen.

Die Option wird an **das Argument**, welches umgerechnet werden soll, in „{}" eingefasst angehängt.

Folgende Konvertierungsoptionen sind z.Z. verfügbar:

Umwandlungen:

%dec	*- wandelt einen HEX-Wert in die entprechende dezimale Zahl um*
%ascii	*- wandelt eine HEX-Sequenz in ASCII Zeichen um*
%hex{Stellen}	*- wandelt eine Dezimalzahl oder eine ASCII Sequenz in eine HEX-Zahl mit {Stellen} (nur bei Dezimalzahlen) um. {Stellen} ist optional.*

Rechnenoperationen:

%-{Wert}	*-subtrahiert {Wert} von Argument*
%+{Wert}	*- addiert {Wert} von Argument*
%/{Wert}	*- dividiert{Wert} von Argument ist Wert = „0" erfolgt keine Berechnung*
%*{Wert}	*- multipliziert {Wert} von Argument*

Die Optionen können auch mehrfach angewendet und zusammengefasst werden. Hierbei ist zu beachten, dass die Berechnung der jeweiligen Werte sich immer auf das Ergebnis der vorherigen Option bezieht.

Zur Verdeutlichung:

<SYS><SETVOL=0A{%dec,%+30,%/2}</SYS> würden den hexadezimalen Wert **0A** in einen dezimalen Wert (10) umwandeln. Der Wert würde nun um 30 addiert werden (40) und wiederum durch 2 dividiert werden. Das Ergebnis 20 würde dann gesendet werden.

Anwendung der der verschiedenen Optionen:

Beispiele:

<SYS><SETVOL=2F{%dec}></SYS> wird zu **<SYS><SETVOL=47></SYS>**
Wandelt den hexadezimalen Wert 2F in die dezimale Zahl 47 um.

<SYS><SAY=414243{%ascii}></SYS> wird zu **<SYS><SAY=ABC></SYS>**
Wandelt die hexadezimalen Werte in die entsprechenden ASCII Zeichen um

<SYS><SAY=15{%hex}></SYS> wird zu **<SYS><SAY=F></SYS>**
Wandelt die dezimale Zahl in eine hexadezimale Zahl um

<SYS><SAY=15{%hex4}></SYS> wird zu **<SYS><SAY=000F></SYS>**
Wandelt die dezimale Zahl in eine hexadezimale Zahl um und erzwingt eine Ausgabe mit 4 Stellen.

<SYS><SAY=A1BC{%hex}></SYS> wird zu **<SYS><SAY=41314243></SYS>**
m..myhome erkennt, dass das Argument nicht nur aus Ziffern besteht und wandelt daher alles in ASCII Zeichencodes um. In diesem Modus wird immer der komplette String umgewandelt. {stellen} hat hier keinen Einfluss.

<SYS><VOLSET=10{%+10}></SYS> wird zu **<SYS><VOLSET=20></SYS>**
Addiert den Wert 10 zu dem Argument

<SYS><VOLSET=10{%\2}></SYS> wird zu **<SYS><VOLSET=5></SYS>**
Dividiert das Argument durch den Wert 2

Komplex:

<SYS><SETVOL=0A{%dec,%+30,%*2,%/4,%-3}></SYS>

wird zu

<SYS><VOLSET=17></SYS>

0A wird in die dezimale Zahl 10 gewandelt. Dieser Wert wird um 30 addiert. Das Ergebnis wird mit 2 multipliziert. Dieses Ergebnis wiederum durch 2 dividiert und das erneute Ergebnis mit 3 subtrahiert.

4.4.3 Parser Optionen (Match Bedingungen).

m..myhome generiert diverse Ausgaben auf die verschiedenen Ports. Diese Ausgaben können, wie im Kapitel 4.4.3 erklärt, automatisch, oder aber z.B. über Mappings Einträge in den CommandServer Definitionen generiert werden.

Um nun diesen Textinhalten gezielt Informationen entnehmen zu können, werden folgende Match Bedingungen/Parser Optionen verwendet:

Erklärung der Zeichen:

\ ^ (\ #)	*Matcht ein Zeichen. Je „^" wird jeweils ein Zeichen erwartet. „#" ist hierbei ebenfalls gültig, sollte aber vermieden werden, da auch als Argument verwendet.*
\ *	*Matcht eine Zeichenkette*
\ ?	*ignoriert ein Zeichen*
\ %	*ignoriert beliebig viele Zeichen*

Die weiteren Erläuterungen und Beispiele für die Verwendung finden Sie in den Kapiteln *CommandServer* und *EventServer*.

4.4.4 Argumente (Übergabe).

Zeichenketten oder Werte, die nach vor genannten Match Bedingungen ausgewertet wurden, werden „gepuffert" und können mit folgenden Argumenten übergeben werden

Erklärung der Zeichen:

\ #	*Übergibt den Inhalt der Matchsequenz bzw den „Puffer"*
\ $	*Übergibt den Inhalt der Matchsequenz bzw den „Puffer" als Liste*
ergänzend zu \ $ **\ &**	*Listenindex nur in Verbindung mit „\ $"*
\ §	*Übergibt die JOIN Nummer* ***(nur mremote.csv)***

Die weiteren Erläuterungen und Beispiele für die Verwendung finden Sie in den Kapiteln *CommandServer* und *EventServer*.

4.4.5 Automatisch generierte Ausgaben (Statusport).

m..myhome generiert automatisch Ausgaben auf den Broadcast Port 1038. So werden automatisch Zustände bzw, Informationen einzelner Applikationen wie zb die aktuelle Lautstärke oder aber der aktuell spielende iTunes Titel generiert. Diese Ausgaben können über den Eventserver ausgewertet und weitere Aktionen angetriggert werden

Folgende Daten kommen über den Broadcast und werden automatisch erzeugt:

ITUNES:	<GETPLAYERSTATE=>	„PLAYING"/"STOPPED"/"PAUSED"
	<GETTITLE=>	„Name des Titels"
	<GETSTREAMTITLE=>	„Name des streaming Titels (IP Radiosender)"
	<GETARTIST=>	„Name des Artisten"
	<GETALBUM=>	„Name des Albums"
	<GETID=>	„Titel ID"
	<GETVOL=>	„Applikationslautstärke"
DVD:	<GETDVDSTATE=>	„PLAYING"/"STOPPED"/"PAUSED"
	<GETTITLE=>	„Name des Titels"
	<GETCHAPTER=>	„Nummer des aktuellen Kapitels"
	<GETVOL=>	„Lautstärke"
TV:	<GETCH=>	„Aktuelle Kanalnunmmer"
	<GETCHNAME=.>	„Aktueller Name des Senders"
	<GETVOL=>	„Applikationslautstärke"
	<GETTUNER=>	„Name des Tuners, der aktuell im Vordergrund"
	<GETPROGINFO=>	mit folgendem Inhalt: „Startzeit aktuelle Sendung" „Endzeit aktuelle Sendung" „Kurzinfo aktuelle Sendung" „Langtext aktuelle Sendung" „Startzeit nächste Sendung" „Endzeit nächste Sendung" „Kurzinfo nächste Sendung" „Langtext nächste Sendung"
SYS:	<GETVOL=>	mit folgendem Inhalt: Audioausgabe „EXT"/"INT" „VOL={0-100}" „MUTE={ON/OFF}"
	<GETSS=>	Status des Bildschirmschoners „ON", „OFF"
Follow-Me-Status:		„FMINFO=...|OK"

xPL	(wenn gültige (= in der .csv definierte) xpl-message empfangen wurde
HS	(immer, wenn etwas empfangen wurde)
KNX	(immer, wenn die Adresse in der .esf definiert ist)
BAOS	(wenn der Datenpunkt in der .csv definiert wurde)
CommandServer	(wie unter „MAPPINGS" definiert)

4.5 Befehlsklassen

m..myhome verfügt über folgende Befehlsklassen:

➢ BROWSER	➢ Steuerung des integrierten Internet- Browsers (Safari)
➢ DVD	➢ Steuerung der Applikation zur DVD-Wiedergabe
➢ ITUNES	➢ Steuerung der Multimedia-Applikation iTunes
➢ SLIDE	➢ Steuerung der integrierten Picture-Slideshow
➢ SCRIPT	➢ Befehlsklasse zur Verwaltung von lokalen Scripten
➢ SCRIPTLOAD	➢ Befehlsklasse zum Laden von Scripten auf den Mac
➢ SYS	➢ applikationsunabhängige Systembefehle
➢ TV	➢ Steuerung der TV-Applikation (EyeTV)
➢ VLC	➢ Steuerung des VideoLAN-Clients
➢ UDP	➢ Senden „freier" UDP Datagramme
➢ TCP	➢ Senden „freier" TCP Pakete
➢ WINDOW	➢ Befehlsklasse zur Anzeige von freien Fensterinhalten (Text/Grafik/Browser)
➢ KNX	➢ KNX/IP - Support
➢ BAOS	➢ BAOS Objektserver - Support
➢ HS	➢ GIRA Homeserver KO-Gateway Support
➢ TIMER	➢ Timer Support

4.6 Kommandolisten

Nachfolgend eine Übersicht über die in den jeweiligen Klassen unterstützten Kommandos:

4.6.1 BROWSER

Kommando	Argument	Beschreibung
ON	-	startet den Browser
OFF	-	beendet den Browser
CLOSEAWIN	-	schließt alle Browserfenster
CLOSEFWIN	-	schließt das letzte Browserfenster
MINFWIN=	{0} oder {1}	minimiert (1) bzw. maximiert (0) das letzte Fenster
MINAWIN=	{0} oder {1}	wie MINFWIN, für alle Fenster
URL=	{URL}	öffnet die angegebene URL in einem neuen Fenster
ZOOM=	{0} oder {1}	zoomt (1) bzw. reduziert (0) das aktuelle Fenster
WFORMAT=	{xmin,ymin,xmax,ymax}	skaliert das aktuelle Fenster auf den angeg. Bereich
SHOW	-	macht die Applikation sichtbar
HIDE	-	versteckt die Applikation
FRONT	-	holt die Applikation nach vorne
WINSIZE=	FULL/{XMAX,YMAX}/{XMIN,YMIN,XMAX,YMAX}	Ändert die Fenstergröße (Ursprung unten links) Erklärung: FULL: streckt das Fenster auf Bildschirmauflösung (soweit die Applikation dies zulässt) XMAX,YMAX: zentriert das Fenster mit der Ausdehnung XMAX,YMAX auf dem Bildschirm XMIN,YMIN,XMAX,YMAX: positioniert das Fenster mit der Ausdehnung XMAX,YMAX und dem Ursprung XMIN,YMIN auf dem Bildschirm

4.6.2 DVD

Kommando	Argument	Beschreibung
ON	-	startet die DVD Player Applikation
OFF	-	beendet die DVD Player Applikation
PLAY	-	spielt die eingelegte DVD ab
STOP	-	stoppt die DVD
PAUSE	-	pausiert die DVD
FF	-	schneller Vorlauf
REW	-	schneller Rücklauf
EJECT	-	wirft die eingelegte DVD aus
URL=	{URL}	öffnet die angegebene URL
FSON	-	aktiviert den Vollbildmodus
FSOFF	-	deaktiviert den Vollbildmodus
MUTEON	-	schaltet den Ton aus
MUTEOFF	-	schaltet den Ton ein
SHOW	-	macht die Applikation sichtbar
HIDE	-	versteckt die Applikation
FRONT	-	holt die Applikation nach vorne
UPKEY	-	simuliert einen Tastendruck (Menü-Navigation)
DNKEY	-	simuliert einen Tastendruck (Menü-Navigation)
LEKEY	-	simuliert einen Tastendruck (Menü-Navigation)
RIKEY	-	simuliert einen Tastendruck (Menü-Navigation)
ENKEY	-	simuliert einen Tastendruck (Menü-Navigation)
GETPTIME	-	liefert die aktuelle Spielzeit in Sekunden
SETPTIME=	{1...x}	springt zur angegebenen Spielzeit
GETCTIME	-	liefert die gesamte Spielzeit in Sekunden
GETCHAPTER	-	liefert das aktuelle Kapitel
SETCHAPTER=	{1...x}	springt zum angegebenen Kapitel
GETALLCHAPTERS	-	liefert die Gesamtzahl aller Kapitel
GETMEDIASTATE	-	liefert den Medien-Status der DVD
GETMENUSTATE	-	liefert den DVD-Menustatus
GETDVDSTATE	-	liefert den Status der DVD
GETCONVIS	-	liefert den Status des Controller-Fensters
SETCONVIS=	{0} oder {1}	(de-)aktiviert das Controller-Fenster
SETCONPOS=	{1...x},{1...y}	setzt die Position des Controller-Fensters
GETINFVIS	-	liefert den Status des Info-Fensters
SETINFVIS=	{0} oder {1}	(de-)aktiviert das Info-Fenster
SETINFPOS=	{1...x},{1...y}	setzt die Position des Info-Fensters

GETTITLE		liest die aktuelle Titelnummer der eingelegten DVD aus
SETTITLE=	{1...x}	springt zur angegebenen Titelnummer
GETDVDTITLE		liefert den aktuellen DVD-Titel
SETVOL= GETVOL WINSIZE=	{0...100} FULL o. {XMAX,YMAX} o. {XMIN,YMIN,XMAX,YMAX}	ändert die Lautstärke des DVD-Players liefert die Lautstärke der DVD-Applikation im Bereich 0...100 FULL oder XMAX,YMAX oder XMIN,YMIN,XMAX,YMAX

4.6.3 ITUNES

Kommando	Argument	Beschreibung
ON	-	startet die iTunes-Applikation
OFF	-	beendet die iTunes-Applikation
PLAY	-	spielt den aktuellen Titel ab
PLAYF=	{Pfad}	spielt die in {Pfad} angegebene Datei ab
STOP	-	stoppt den aktuellen Titel
PAUSE	-	pausiert den aktuellen Titel
FF	-	schneller Vorlauf
REW	-	schneller Rücklauf
NEXT	-	springt zum nächsten Titel
PREV	-	springt zum vorigen Titel
RESUME	-	setzt die Wiedergabe fort
ADD=	{Pfad}	fügt eine Datei zur aktuellen Playliste hinzu
URL=	{URL}	öffnet die angegebene URL (Streaming)
WFORMAT=	xmin,ymin,xmax,ymax	skaliert das aktuelle Fenster auf den angeg. Bereich
SHOW	-	macht die Applikation sichtbar
HIDE	-	versteckt die Applikation
FRONT	-	holt die Applikation nach vorne
SHOWPLAYER	-	*blendet das iTunes-Hauptfenster ein
HIDEPLAYER	-	*blendet das iTunes-Hauptfenster aus
GETVOL	-	liefert die aktuelle Applikationslautstärke
SETVOL=	{1...100}	setzt die Applikationslautstärke
MUTEON	-	schaltet den Ton aus
MUTEOFF	-	schaltet den Ton ein
GETMUTE	-	liefert den Mute-Status
GETPLAYLIST	-	liefert die aktuelle Playliste
GETALLPLAYLISTS	-	liefert eine Liste aller vorhanden Playlisten
GETTITLE	-	liefert den aktuellen Titel
GETARTIST	-	liefert den Künstler des aktuellen Titels
GETALBUM	-	liefert das Album des aktuellen Titels
GETPLAYLISTCOUNT		liefert die Anzahl der vorhandenen Playlisten
GETID		liefert die ID des aktuellen Titels
GETALLIDSOFPLAYLIST=	{Playlist}	liefert alle IDs in {Playliste}
GETSTREAMTITLE	-	liefert den Titel des aktuellen Streams
PLAYPLAYLIST=	{Name}	spielt die Playliste {Name} ab
GETPTIME	-	liefert die aktuelle Spielzeit in Sekunden
SETPTIME=	{1...x}	springt zur angegebenen Spielzeit
GETCTIME	-	liefert die gesamte Spielzeit in Sekunden

GETVISSTATE	-	liefert den Visualisierungs-Status
SETVISSTATE=	„0" oder „1"	(de-)aktiviert die Visualisierungen
GETPLAYERSTATE	-	liefert den Status des Players
ENABLECF	-	Startet das iTunes Cover Flow im Vollbildmodus
DISABLECF	-	Beendet das iTunes Cover Flow
SEARCHTITLE=	{Suchstring}	liefert eine Track-ID -Liste der Titel zurück, auf die der Suchstring paßt (Limit: 500)
SEARCHARTIST=	{Suchstring}	liefert eine Track-ID -Liste der Interpreten zurück, auf die der Suchstring paßt (Limit: 500)
SEARCHALBUM=	{Suchstring}	liefert eine Track-ID -Liste der Alben zurück, auf die der Suchstring paßt (Limit: 500)
GETIDTITLE=	{ID}	liefert den Titel zur angegebenen Track-ID
GETIDARTIST=	{ID}	liefert den Interpreten zur angegebenen Track-ID
GETIDALBUM=	{ID}	liefert das Album zur angegebenen Track-ID
PLAYID=	{ID}	spielt den Titel {ID} ab
CREATEPLAYLIST=	{NAME}	erzeugt eine neue Playlist {NAME}
DELETEPLAYLIST=	{NAME}	löscht die Playlist {NAME}
ADDIDTOPLAYLIST=	{ID},{NAME}	fügt Titel {ID} zur Playlist {NAME} hinzu
WINSIZE=	FULL/{XMAX,YMAX}/{XMIN,YMIN,XMAX,YMAX}	Ändert die Fenstergröße (Ursprung unten links) Erklärung: FULL: streckt das Fenster auf Bildschirmauflösung (soweit die Applikation dies zulässt) XMAX,YMAX: zentriert das Fenster mit der Ausdehnung XMAX,YMAX auf dem Bildschirm XMIN,YMIN,XMAX,YMAX: positioniert das Fenster mit der Ausdehnung XMAX,YMAX und dem Ursprung XMIN,YMIN auf dem Bildschirm
SETSHUFFLE#	{ON/OFF}	schaltet den Modus Shuffle der aktuellen Playlist ein o. aus
GETSHUFFLE		gibt den aktuellen Playmodus der Playliste zurück
SETREPEAT#	{ONE/ALL/OFF}	setzt den Repeatmodus der aktuellen Playlist

4.6.4 SLIDE

Kommando	Argument	Beschreibung
ON	-	startet die iPhoto-Applikation
OFF	-	beendet die iPhoto-Applikation
START	-	startet die Slideshow
STOP	-	beendet die Slideshow
PAUSE	-	pausiert die Slideshow
RESUME	-	setzt die Slideshow fort
PREV	-	springt zum vorigen Bild
NEXT	-	springt zum nächsten Bild
IMPORT=	{Pfad}	importiert die Bilder im angegebenen Pfad
SHOW	-	macht die Applikation sichtbar
HIDE	-	versteckt die Applikation
FRONT	-	holt die Applikation nach vorne
GETALBUMLIST	-	liefert eine Liste aller Alben
SELALBUM=	{Name.suffix}	selektiert das Album {Name.suffix}
GETALBUM	-	liefert den Namen des aktuellen Albums

4.6.5 SCRIPT

Kommando	Argument	Beschreibung
RUN=	{Name.suffix}	führt das MyH-Script {Name.suffix} aus
ARG=	{Wert}	Übergibt einen oder mehre variablen Wert an das Script. (füllt sequentiell den o. die Platzhalter ARG)
RUNAS=	{Name.suffix}	führt das Applescript {Name.suffix} aus
RUNSS=	{Name.suffix}	führt das Shell-Script {Name.suffix} aus
DELETE=	{Name.suffix}	löscht das Script {Name.suffix}
RENAME=	{Name.suffix}={Name_neu.suffix}	benennt myh Script {Name.suffix} in {Name_neu.suffix} um
LIST	-	listet alle MyH-Scripte im scripts-Ordner
LISTAS	-	listet alle Applescripte im scripts-Ordner
LISTSS	-	listet alle Shell-Scripte im scripts-Ordner
SHOW=	{Name.suffix}	zeigt den Inhalt des Scripts {Name}

4.6.6 SCRIPTLOAD

Kommando	Argument	Beschreibung
NAME=	{Name.suffix}	legt ein neues Script {Name.myh} an, alle nachfolgenden Klassen und Argumente definieren den Inhalt des myh Scripts, bis der Klassen-Terminator </SCRIPTLOAD> gelesen wurde

4.6.7 SYS

Kommando	Argument	Beschreibung
SETVOL=	{0...100}	Setzt die Systemlautstärke auf 0-100%
VOLUP=	{0...100}	erhöht die Systemlautstärke um 0-100%
VOLDN=	{0...100}	verringert die Systemlautstärke um 0-100%
MUTEON	-	schaltet den Systemton ab
MUTEOFF	-	schaltet den Systemton ein
GETVOL	-	liefert die Systemlautstärke
GETSTATE	-	liefert Informationen über den Applikations-Status
UPKEY	-	simuliert den Tastendruck „Cursor hoch"
DNKEY	-	simuliert den Tastendruck „Cursor runter"
LEKEY	-	simuliert den Tastendruck „Cursor links"
RIKEY	-	simuliert den Tastendruck „Cursor rechts"
ENKEY	-	simuliert den Tastendruck „Enter"
SPCKEY	-	simuliert den Tastendruck „Leertaste"
TABKEY	-	simuliert den Tastendruck „Tabulator"
PUSHUPKEY		simuliert den "hold" Tastendruck „Cursor hoch"
PUSHDNKEY		simuliert den "hold" Tastendruck „Cursor runter"
PUSHLEKEY		simuliert den "hold" Tastendruck „Cursor links"
PUSHRIKEY		simuliert den "hold" Tastendruck „Cursor rechts"
SETMPOS=	{0...x},{0...y}	setzt den Mauszeiger auf x,y
MCLICK=	{0...x},{0...y} o. HERE	{x,y} oder HERE - simuliert einen Mausklick an der Position x,y, ohne den Mauszeiger zu bewegen. HERE klickt an der aktuellen Position des Mauszeigers
GETMPOS		liefert die aktuellen Koordinaten des Mauszeigers
ENABLEFR	-	aktiviert die FrontRow-Applikation
DISABLEFR	-	deaktiviert die FrontRow-Applikation
ENABLESS	-	aktiviert den Bildschirmschoner
DISABLESS	-	deaktiviert den Bildschirmschoner
ENABLEDB	-	aktiviert das Dashboard
DISABLEDB	-	deaktiviert das Dashboard
SHOWDOCK	-	zeigt das Dock
HIDEDOCK	-	versteckt das Dock
HIDEALL	-	Bringt alle Anwendungen in den Hintergrund (Hide)
HIDESTOPFRONT	-	versteckt, stoppt und mutet die oberste Applikation
OPEN=	{Name.suffix}	versucht, die Datei {Name.suffix} zu öffnen
SHUTDOWN	-	fährt das System herunter
RESTART	-	startet das System neu
SLEEP	-	versetzt das System in den Ruhezustand
HELLO	-	liefert Informationen über den M..myhome

GETLICENSEINFO	-	liefert Informationen über die aktuell installierte Lizenz. Format der Antwort: {Gültigkeit}|{Commandserver}|{Event-Support}|{xPL-Support}|{BAOS-Support}|{Extended-Buttons-Support}
FADE=	{1} oder {0}	blendet den Bildschirm aus bzw. ein.
MESSAGE=	{„text"}	Blendet den Bildschirm aus und zeigt den Text {text} zentriert auf dem Bildschirm an. Ein „|" im Text-String erzeugt einen Zeilenvorschub. Entfällt in späteren Versionen.
MESSAGEOFF=		Blendet den mit MESSAGE angezeigten Text aus und zeigt den Desktop an. Entfällt in späteren Versionen.
SAY=	{Text}	Übergibt einen Text an die Sprachausgabe
RESTARTSERVICE		Startet und resettet den Daemon neu.
CMDKEY=#	{Keycode}	Simuliert Tastendruck "Apfel"+{Keycode}. Wichtig: Es wird der Apple- Tastaturcode erwartet, nicht der ASCII-Code! Eine entsprechende Tabelle finden Sie im Anhang der Dokumentation
ALTCMDKEY=	{Keycode}	Simuliert Tastendruck „ALT" + "Apfel"+{Keycode}. Wichtig: Es wird der Apple- Tastaturcode erwartet, nicht der ASCII-Code! Eine entsprechende Tabelle finden Sie im Anhang der Dokumentation
CTRLCMDKEY#	{Keycode}	Simuliert Tastendruck „CTRL" + "Apfel"+{Keycode}. Wichtig: Es wird der Apple- Tastaturcode erwartet, nicht der ASCII-Code! Eine entsprechende Tabelle finden Sie im Anhang der Dokumentation
SHIFTCMDKEY#	{Keycode}	Simuliert Tastendruck "SHIFT" + "CMD"+{Keycode}
KEY#	{Keycode}	Simuliert Tastendruck. Wichtig: Es wird der Apple-Tastaturcode erwartet, nicht der ASCII-Code! Eine entsprechende Tabelle finden Sie im Anhang der Dokumentation
FMINIT	-	Initialisiert den Follow-me Status
FMREAD	-	liefert den generierten Follow-me String. (Setzt ein erfolgreiches vorangegangenes FMINIT voraus.)
FMEXIT	-	verlässt den Follow-me Status. (Setzt ein erfolgreiches vorangegangenes FMINIT voraus.)
FMABORT	-	bricht den Follow-me ab und setzt die Wiedergabe fort. (Setzt ein erfolgreiches vorangegangenes FMINIT voraus.)
FMINFO	-	liefert den aktuellen Follow-me Status (TV,DVD,...) (auch im Broadcast vorhanden)
GETSRVLIST	-	interner Befehl für das Einleiten zum Auslesen der AddOn Befehle. Nach Auslösen wird das Menu um evtl. vorhandene AddOn's erweitert
GETXPLLIST	-	interner Befehl für das Einleiten zum Auslesen der XPL Befehle. Nach Auslösen wird das Menu um evtl. vorhandene AddOn's erweitert
GETFILENUMINFOLDER=	{Pfad}	liefert die Zahl der in {Pfad} vorhandenen Dateien
GETFOLDERNUMINFOLDER=	{Pfad}	liefert die Zahl der in {Pfad} vorhandenen Verzeichnisse
GETFILESINFOLDER=	{Pfad,Index=x,Length=y}	liefert eine Liste der in {Pfad} vorhandenen Dateien, Index und Length sind optional
GETFOLDERSINFOLDER=	{Pfad,Index=x,Length=y}	liefert eine Liste der in {Pfad} vorhandenen Verzeichnisse, Index und Length sind optional

OPENBUTTONS=	{*ButtonLeistenName*}	öffnet eine Button Leiste
CLOSEBUTTONS=	{*ButtonLeistenName*}	Schließt eine Button Leiste
SETBUTTONPUSHED=	{*ButtonLeistenName*}	setzt den Zustand eines Buttons auf betätigt (wenn Toggle Button)
SETBUTTONRELEASED=	{*ButtonLeistenName*}	setzt den Zustand eines Buttons auf nicht betätigt (wenn Toggle Button)
CLOSECURRENTBUTTONS		schließt die aktuell geöffnete Buttonleiste
SETBUTTONLABEL=	{*ButtonLeistenName*}, Index {1-10},{Label Text}	Verändert flüchtig das Label des Buttons
SETALTBUTTONLABEL=	{*ButtonLeistenName*}, Index {1-10},{Label Text}	Verändert flüchtig das 2. Label eines SWITCH Buttons
SHOWKEYBOARD		Blendet die Eingabehilfe ein (Keyboard)
HIDEKEYBOARD		Blendet die Eingabehilfe aus (Keyboard)
GETVAR=	{Variablenname}	liest den Wert einer System-Variablen
SETVAR=	{Variablenname}	ändert den Wert einer System-Variablen bzw. legt sie an
DELVAR=	{Variablenname}	löscht eine System-Variable
GETALLVARS		Abfrage und Ausgabe der Werte aller vorhandenen Variablen
GETSERIALPORTLIST		listet alle im System vorhandene, serielle Ports auf
GETSPEAKERLIST		listet alle verfügbaren Airports mit Namen auf ("Computer" ist der lokale Ton des Macs)
GETSPEAKERSOURCE		gibt die aktuelle Applikation bzw. den aktiven Eingang aus, dessen Ton gerade gestreamt wird
SETSPEAKERSOURCE=	{Appname}	wählt die Applikation aus, die gestreamt werden soll
SETSPEAKERSOURCEDEVICE=	{Eingang}	wählt den Audio-Eingang aus, der gestreamt werden soll
GETAUDIODEVICELIST		listet alle verfügbaren Audio-Eingänge auf
GETSPEAKERVOL=	{Airport-Name}	{Airport-Name}
SETSPEAKERVOL=	{Airport-Name},{0...100}	setzt die Lautstärke des gewählten Airports absolut, wird der Name weggelassen, wird die Lautstärke für alle Airports (inkl. "Computer") gesetzt
SPEAKERVOLUP=	{Airport-Name},{0...100}	erhöht die Lautstärke um den angegebenen Wert, der Name ist auch hier optional
SPEAKERVOLDN=	{Airport-Name},{0...100}	senkt die Lautstärke, sonst wie SPEAKERVOLUP
ENABLESPEAKER=	{Airport-Name}	schaltet den gewählten Airport ein
DISABLESPEAKER=	{Airport-Name}	schaltet den gewählten Airport aus
ENABLEALLSPEAKERS		schaltet alle Airports ein
DISABLEALLSPEAKERS		schaltet alle Airports aus
GETSPEAKERSTATE=	{Airport-Name}	liefert den Status des gewählten Airports: ENABLED / DISABLED / NOT_AVAILABLE
SETDIGITALJOIN=	{Nummer},{Wert}	mRemote Support - setze digital Join (0/1)
SETANALOGJOIN=	{Nummer},{Wert}	mRemote Support - setze analog Join (0-65535)
SETSERIALJOIN=	{Nummer},{Wert}	mRemote Support - setze serial Join (Text)
WAKEONLAN=	{MAC-Adresse}	sendet ein Magic Packet für die angegebene MAC-Adresse. Die Schreibweise der Mac-Adresse ist dabei egal, solange exakt 6 Octets angegeben werden.

GETAUDIOOUTPUTLIST		listet alle verfügbaren Audio-Ausgabegeräte mit Namen auf
GETAUDIOOUTPUT		liefert das aktuelle Ausgabegerät
SETAUDIOOUTPUT=	{Name}	schaltet auf Ausgabegerät {Name} um
SETDISPLAYMODE=	{x,y,bpp}, {x,x}, "0" oder "NORMAL"	Ändert die Bildschirmauflösung, wenn unterstützt {x,y,bpp}, {x,x}, "0" oder "NORMAL"
GETTIME		liefert die aktuelle Systemzeit im KNX EIS3 Format
GETDATE		liefert das aktuelle Systemdatum im KNX EIS3 Format
SETCOUNTER=	{Name}{,Startwert}{,Schrittweite}	initialisiert den Counter {Name}.{Startwert} und {Schrittweite} sind optional.Default ist 0 für {Startwert} und 1 für {Schrittweite}
SETCOUNTERSTEP=	{Name}{,Schrittweite}	ändert die Schrittweite eines bestehenden Counters {Name}, 0 für {Schrittweite} hält den Counter an.
GETCOUNTER=	{Name}	liefert den aktuellen Wert des Counters {Name}, ohne sie zu verändern
GETALLCOUNTERS		liefert die aktuellen Werte aller Counter, ohne diese zu verändern

Hinweis zum SETDISPLAYMODE-Befehl:

Wenn die Angaben nicht umsetzbar sind (Auflösung und/oder Farbtiefe), wird automatisch der nächstmögliche Modus eingestellt (siehe Log) Es ist nicht möglich, Modi einzustellen, die vom Bildschirm nicht unterstützt werden. Beim Beenden von mmh wird der Bildschirm auf den ursprünglichen Modus zurückgesetzt.

4.6.8 TV

Kommando	Argument	Beschreibung
SETVOL=	{0...100}	Setzt die Systemlautstärke (TV) auf 0-100%
ON	-	startet die TV-Applikation
OFF	-	beendet die TV-Applikation
PLAY	-	startet die Wiedergabe
PAUSE	-	pausiert die Wiedergabe
STOP	-	stoppt die Wiedergabe
FF	-	schneller Vorlauf
REW	-	schneller Rücklauf
SLOWF	-	Zeitlupe vorwärts
SLOWR	-	Zeitlupe rückwärts
SKIPF	-	springt nach vorne
SKIPR	-	springt zurück
RECSTART	-	startet die Aufnahme
RECSTOP	-	beendet die Aufnahme
CHUP	-	schaltet einen Kanal nach oben
CHDN	-	schaltet einen Kanal nach unten
FSON	-	aktiviert den Vollbildmodus
FSOFF	-	deaktiviert den Vollbildmodus
ZOOM=	{1} oder {0}	zoomt (1) bzw. reduziert (0) das aktuelle Fenster
WFORMAT=	{xmin,ymin,xmax,ymax}	skaliert das aktuelle Fenster auf den angeg. Bereich
SHOW	-	macht die Applikation sichtbar
HIDE	-	versteckt die Applikation
FRONT	-	holt die Applikation nach vorne
HIDECON	-	versteckt das Konsolenfenster
SHOWCON	-	zeigt die Konsolenfenster
CH=	{1...x}	wählt den Kanal
GETCH	-	liefert die aktuelle Kanalnummer
GETCHNAME=	{1...x}	liefert den Kanalnamen zur Kanalnummer
MUTEON	-	schaltet den Applikationston aus
MUTEOFF	-	schaltet den Applikationston ein
ENABLEEGP	-	Schaltet die EPG Ansicht ein
DISABLEEPG	-	Schaltet die EPG Ansicht aus
GETPROGINFO		liefert Informationen über die aktuelle und die nächste Sendung im Format: {Start}|{Stop}|{Titel der aktuellen Sendung}|{Kurzbeschreibung}| {Start}|{Stop}|{Titel der nächsten Sendung}|{Kurzbeschreibung}.
WINSIZE=	FULL/{XMAX,YMAX}/{XMIN ,YMIN,XMAX,YMAX}	Ändert die Fenstergröße (Ursprung unten links) Erklärung: FULL: streckt das Fenster auf Bildschirmauflösung

		(soweit die Applikation dies zulässt) XMAX,YMAX: zentriert das Fenster mit der Ausdehnung XMAX,YMAX auf dem Bildschirm XMIN,YMIN,XMAX,YMAX: positioniert das Fenster mit der Ausdehnung XMAX,YMAX und dem Ursprung XMIN,YMIN auf dem Bildschirm
ENABLEMENU		blendet das eyeTV-OSD-Menu ein
DISABLEMENU		blendet das eyeTV-OSD-Menu aus
ENABLEFSEPG		blendet die Vollbild EPG Ansicht ein
DISABLEFSEPG		blendet die Vollbild die EPG Ansicht aus

4.6.9 VLC

Kommando	Argument	Beschreibung
ON	-	aktiviert den VideoLAN-Client
OFF	-	beendet den VideoLAN-Client
URL=	{URL}	öffnet die Datei/den Stream {URL}
TPLAY	-	toggelt den Play-Status
STOP	-	stoppt die Wiedergabe
PREV	-	springt zum vorigen Titel
NEXT	-	springt zum nächsten Titel
TFS	-	toggelt den Vollbild-Modus
TMUTE	-	toggelt den Mute-Status
VOLUP	-	erhöht die Lautstärke
VOLDN	-	reduziert die Lautstärke
ZOOM=	{1} oder {0}	zoomt (1) bzw. reduziert (0) das aktuelle Fenster
WFORMAT=	{xmin,ymin,xmax,ymax}	skaliert das aktuelle Fenster auf den angeg. Bereich
SHOW	-	macht die Applikation sichtbar
HIDE	-	versteckt die Applikation
FRONT	-	holt die Applikation nach vorne
WINSIZE=	FULL/{XMAX,YMAX}/{XMIN,YMIN,XMAX,YMAX}	Ändert die Fenstergröße (Ursprung unten links) Erklärung: FULL: streckt das Fenster auf Bildschirmauflösung (soweit die Applikation dies zulässt) XMAX,YMAX: zentriert das Fenster mit der Ausdehnung XMAX,YMAX auf dem Bildschirm XMIN,YMIN,XMAX,YMAX: positioniert das Fenster mit der Ausdehnung XMAX,YMAX und dem Ursprung XMIN,YMIN auf dem Bildschirm

4.6.10 UDP/TCP

Beispiel:

<UDP><IP=192.168.50.1><PORT=4000><MODE=HEX><TIME=5><DATA=414243></UDP>

Sendet ein UDP-Paket mit dem Inhalt "ABC" an die IP 192.168.50.1 auf Port 4000 und wartet 5 Sekunden auf Antwort(en). Innerhalb des Timeouts empfangene Daten werden sofort weitergereicht, bis der Timeout abgelaufen ist. Wird kein Timeout angegeben (oder TIME=0), wird das Paket nur versendet, aber keine Antwort abgewartet. Der maximal mögliche Timeout-Wert ist auf 3600 festgesetzt.

<UDP><IP=192.168.50.125><PORT=1037><MODE=ASCII><TIME=0><DATA=<SYS><MUTE></SYS></UDP>

Sendet den Befehl **System Mute** an das "entfernte" m..myhome System mit der Adresse 192.168.50.125

Nach Absetzen eines UDP-Paketes kehrt der Daemon sofort wieder in den Kommandomodus zurück und verarbeitet evtl. weitere anstehende Kommandos. Das Handling der UDP-Verbindungen läuft parallel.

Die Klasse TCP ist konform der Klasse UDP.

Kommando	Argument	Beschreibung
IP=	{IP Adresse}x	IP Adresse des Empfängers in Punktnotation
PORT=	{PORT}	Port Adresse des Empfängers
MODE=	{„HEX","ASCII","BIN"}	legt den Übertragungsmodus fest (z.Z. nur HEX)
TIME=	{t in Sekunden 0 - 3600}	optional, legt den Timeout in Sekunden fest, bis zu dem auf eine Antwort gewartet wird
DATA=	{Daten}	Dateninhalt des UDP-Paketes im MODE-Format

<TCP><IP=192.168.50.1><PORT=4000><MODE=HEX><TIME=5><DATA=414243></TCP>

Sendet ein TCP Paket mit dem Inhalt "ABC" an die IP 192.168.50.1 auf Port 4000 und wartet 5 Sekunden auf Antwort(en). Innerhalb des Timeouts empfangene Daten werden sofort weitergereicht, bis der Timeout abgelaufen ist. Wird kein Timeout angegeben (oder TIME=0), wird das Paket nur versendet, aber keine Antwort abgewartet. Der maximal mögliche Timeout-Wert ist auf 3600 festgesetzt.

<TCP><IP=192.168.50.125><PORT=1037><MODE=ASCII><TIME=0><DATA=<SYS><MUTE></SYS></TCP>

Sendet den Befehl **System Mute** an das "entfernte" m..myhome System mit der Adresse 192.168.50.125

4.6.11 Window

Bei der Verwendung der WINDOWS Klasse ist folgendes zu beachten:

WINDOWS werden aus dem SAFARI Browser im Kioskmode (Frameless) generiert. Folglich können in dieser Klasse alle Inhalte generiert werden, die sich mit dem SAFARI Browser und deren mögliche PlugIn's anzeigen und steuern lassen.

Ebenso zu beachten ist das „Schichtenmodel" (LAYER). Hierbei werden die WINDOW'S nach Prioritäten behandelt. Somit kann gezielt gesteuert werden, welches WINDOW an „oberster" Stelle zur Anzeige gebracht wird. Das Schichtenmodel unterscheidet hierbei zwei Prioritätsgruppen. Dies definiert sich aus dem LEVEL und aus dem SELECT Befehl.

Wenn LEVEL nicht angegeben wird, wird automatisch LEVEL=NORMAL angenommen. Durch Level wird der "Raum", in dem sich das Fenster befindet, ausgewählt. Die Auswahl der Fensterreihenfolge innerhalb des "Raums" kann nach wie vor mit Hilfe des SELECT-Befehls erfolgen. Ein Fenster mit SELECT=2 liegt also über einem Fenster mit SELECT=1, wenn beide das selbe LEVEL besitzen. Entsprechend liegt ein Fenster mit SELECT=3 und LEVEL=HIGH über(!) einem Fenster mit SELECT=4 und LEVEL=NORMAL, obwohl die ID des Fensters niedriger ist.

Der LEVEL-Befehl kann jederzeit - auch bei bereits dargestellten Fenstern - angewendet werden.

Kommando	Argument	Beschreibung
XMIN#	{0...}	X-Koordinate der unteren linken Ecke des Windows (zwingend), **default=0**
YMIN#	{0...}	Y-Koordinate der unteren linken Ecke des Windows (zwingend)
XMAX#	{0...}	X-Koordinate der oberen rechten Ecke des Windows (zwingend), **default=max**
YMAX#	{0...}	Y-Koordinate der oberen rechten Ecke des Windows (zwingend)
LEVEL=	{LOW,NORMAL,HIGH}	NORMAL (default): Window über allen normalen Fenstern, unterhalb des Bildschirmschoners. HIGH: über dem Bildschirmschoner LOW: Window unter allen Fenstern, auf dem Desktop-Hintergrund
Ergänzung zur Verwendung von X/YMIN/MAX		Wenn XMIN und YMIN in einer Fensterkonfiguration weggelassen werden, wird das Fenster automatisch zentriert dargestellt. XMIN und XMAX sind mit default-Werten zur Bannerdarstellung in Bildschirmbreite vorbesetzt
SHADOW#	„0" oder „1"	Schatten zeichnen?
TRANS#	{0.0...1.0}	Transparenz des Fensters (zwingend) - Dieser Befehl kann auch nachträglich auf ein bereits dargestelltes Fenster angewendet werden, z.B. zum langsamen Einblenden. **default=0.9**
TITLE#	{Text}	Titel des Fensters
URL#	{Text}	öffnet die URL im Fenster - keine weiteren Kommandos notwendig
IMAGE#	{Text(URL/Pfad)}	zeigt ein lokales oder entferntes Bild an. Das Bild kann entweder als Pfad (z.B. "/Users/MyH/...") oder als URL ("http://.....") angegeben werden. Die Regeln für die Darstellung des Bildes gelten analog zum URL-Befehl
TEXT#	{Text}	stellt TEXT zentriert im Fenster dar – weitere Kommandos notwendig. eine Pipe im darzustellenden Text erzeugt einen Zeilenvorschub an der entsprechenden Stelle. Die momentan voreingestellte

		Schriftart für alle Textdarstellungen ist Helvetica-Bold Oblique
TEXTSIZE#	{0.0...1.0}	Textgröße in Pixel (zwingend)
TEXT_RED#	{0.0...1.0}	Rot-Anteil der Textfarbe - default: 1.0
TEXT_GREEN#	{0.0...1.0}	Grün-Anteil der Textfarbe - default: 1.0
TEXT_BLUE#	{0.0...1.0}	Blau-Anteil der Textfarbe - default: 1.0
TEXT_TRANS#	{0.0...1.0}	Transparent-Anteil der Textfarbe - default:1.0
BACKGROUND_RED#	{0.0...1.0}	Rot-Anteil des Texthintergrundes - default: 0
BACKGROUND_GREEN#	{0.0...1.0}	Grün-Anteil des Texthintergrundes - default: 0
BACKGROUND_BLUE#	{0.0...1.0}	Blau-Anteil des Texthintergrundes - default: 0
BACKGROUND_TRANS#	{0.0...1.0}	Transparent-Anteil des Texthintergrundes - default: 1.0
FADETO#	{0..1.0}	blendet die aktuelle Transparenz des Fensters langsam zum angegebenen Wert
CREATE		Abschluss der Konfiguration: erzeugt das Fenster auf dem Bildschirm, wenn nicht geöffnet. (zwingend)
RELEASE		Entfernt das Fenster vom Bildschirm
MINIMIZE		Fenster im Dock minimieren
MAXIMIZE		Fenster im Dock wiederherstellen
SELECT#	{1…10}	Fensterauswahl (1-10) Alle nachfolgenden Befehle beziehen sich immer auf das zuvor mit SELECT gewählte Fenster. SELECT ist optional (Defaultwert:1). Je höher der Wert, desto höher die Priorität. Ein WINDOW mit SELECT=2 liegt also über dem WINDOW mit SELECT=1
RELEASEALL		schließt alle Fenster und setzt SELECT auf 1
RESET		löscht alle Fenster und setzt sämtliche Parameter auf 0
EFFECTCREATE		wie CREATE, aber mit Animation
EFFECTRELEASE		wie RELEASE, aber mit Animation
MOVE		bewegt ein bereits dargestelltes Fenster zu den neuen Koordinaten, die zwischenzeitlich gesetzt wurden
EFFECTMOVE		wie MOVE, aber mit zusätzlicher Animation

4.6.12 BAOS

Kommando	Argument	Beschreibung
GETINFO	-	liefert allgemeine Informationen über den BAOS
GETDESCRIPTION=	{Datenpunktnummer}	liefert die Einstellungen des gewählten Datenpunktes
GETDESCRIPTIONSTRING=	{Datenpunktnummer}	liefert die Beschreibung des gewählten Datenpunktes
GETVALUE=	{Datenpunktnummer}	liefert den aktuellen Wert des Datenpunktes inkl. Zustandsflags im HEX-Format
SETVALUE=	{Datenpunktnummer},{HEX/DEC/ASCII}	setzt den Wert des gewählten Datenpunkt in HEX oder als ASCI
SENDVALUE=	{Datenpunktnummer},{HEX/DEC/ASCII}	Sendet den Wert auf den KNX-Bus
SETANDSENDVALUE=	{Datenpunktnummer},{HEX/DEC/ASCII}	Kombination von SETVALUE und SENDVALUE
READVALUE=	{DatenpunktNr.}	liest den Wert vom KNX-Bus
CLEARTRANSMISSIONSTATE=	{Datenpunktnummer},{HEX/DEC/ASCII}	löscht den Verbindungsstatus
GETPARAMETERBYTE=	{ParameterNr.)	liefert das Byte des gewählten Parameters

4.6.13 HS

Kommando	Argument	Beschreibung
SETVALUE=	{Adresse},{Wert}	setzt {Wert} an {Adresse} absolut
SETVALUEREL=	{Adresse},{Wert}	setzt {Wert} an {Adresse} relativ
STEPUP=	{Adresse},{Wert}	erhöht den Wert an {Adresse}
STEPDOWN=	{Adresse},{Wert}	senkt den Wert an {Adresse}
LISTUP=	{Adresse},{Wert}	erhöht den Wert an {Adresse}
LISTDOWN=	{Adresse},{Wert}	senkt den Wert an {Adresse}

4.6.14 Streamer

Kommando	Argument	Beschreibung
ON		startet die Streaming-Engine und ggfs EyeTV (minimiert)
OFF		beendet die Streaming-Engine, EyeTV läuft weiter
CH=	{Kanalnummer} oder {Kanalname}	wählt die Streaming-Quelle, dieser Befehle funktioniert analog zum SELECT-Befehl in der Windows- Klasse. Alle nachfolgenden, kanalbezogenen Befehle beziehen sich auf den zuletzt mit CH ausgewählen Kanal
GETCHLIST		liefert eine Liste mit allen zur Zeit verfügbaren Kanälen und der entsprechenden Kanalnummer
ADDTARGET=	{IP:Port}	weist der zuvor mit CH gewählten Quelle ein Ziel zu und beginnt sofort mit dem UDP-Streaming zur angegebenen Adresse. es wird eine eindeutige Streaming-ID zurückgegeben. Auf dem Client dem VLC bitte als URL: udp://@:{Portnummer} übergeben.
REMOVETARGET=	{IP:Port} oder {Streaming-ID} oder {IP}	stoppt das Streaming des per IP:Port oder ID angegeben Streams. Wird nur die IP angegeben, werden alle an diese IP gerichteten Streams gestoppt.
REMOVEALLTARGETS		stoppt alle Streams
GETTARGETSFORIP=	{IP}	listet alle an die IP gerichteten Streams auf
GETTARGETSFORCH=	{Kanalnummer} oder {Kanalname}	listet alle IP- Adressen auf, an die der Kanal gestreamt wird
GETALLTARGETS		listet alles auf, was gestreamt wird

4.6.15 KNX

Zur Verwendung der KNX Klasse müssen die Datenpunkte, die angesprochen werden sollen, in der Definitionsdatei „**misc/**knx.esf" vorhanden sein.

Kommando	Argument	Beschreibung
SETVALUE=	{Adresse},{Wert}	setzt {Wert} an {Adresse}
GETVALUE=	{Adresse}	Liest einen Wert einer {Adresse}

4.6.16 TIMER

Timerdefinitionen müssen in der Datei misc/timer.csv definiert sein.

Kommando	Argument	Beschreibung
START=	{Name}	startet den Timer {Name}
STOP=	{Name}	stoppt den Timer {Name}
LIST=		listet alle aktiven Timer auf

4.6.17 Sonderkommandos

Sonderkommandos sollten nicht innerhalb einer geöffneten Klasse verwendet werden. Diese Befehle sollten immer nach Abschluss- und vor dem Öffnen einer Klasse platziert werden.

Kommando	Argument	Beschreibung
DELAY=	{t in Sekunden(1 für 1 Sekunde oder 0.5 für 500ms)}	verzögert die Ausführung der nachfolgenden Befehle um x Sekunden. (Klassenlos) Bitte immer außerhalb einer Klassensequenz verwenden.
RELAY=	{IP}:{Port}	Die mmh-Sequenz wird per UDP an einen entfernten Mac weitergeleitet und maximal 3 Sekunden auf eine Antwort gewartet. Beispiel: <RELAY=192.168.1.50:1037><SYS><ENABLESS></SYS></RELAY> Die Antwortsequenz hat die Syntax: <RELAY={IP}:{Port}>{mmh-Antwortsequenz}</RELAY>
BREAK		beendet sämtliche z.Z. laufenden Kommandosequenzen. (Klassenlos)

4.7 Tastaturbelegung (Sondertasten)

Für die Bearbeitung der folgend beschriebenen Definitionstabellen, werden teilweise Zeichen benötigt, die auf dem Tastaturlayout des Mac Systems nicht vorhanden sind. Nachfolgende Tabelle liefert einen Auszug der notwendigen und wichtigen Zeichen sowie deren Aufruf:

@	Mac OS = **[ALT] + L**	**@**	Win OS = **[CTRL] + [ALT] + Q**
|	Mac OS = **[ALT] + 7**	**|**	Win OS = **[CTRL] + [ALT] + <**
[	Mac OS = **[ALT] + 5**	**[**	Win OS = **[CTRL] + [ALT] +8**
]	Mac OS = **[ALT] + 6**	**]**	Win OS = **[CTRL] + [ALT]+ 9**
{	Mac OS = **[ALT] + 8**	**{**	Win OS = **[CTRL] + [ALT]+ 7**
}	Mac OS = **[ALT] + 9**	**}**	Win OS = **[CTRL] + [ALT]+ 0**
****	Mac OS = **[ALT] + [SHIFT] + 7**	****	Win OS = **[CTRL] + [ALT]+ ß**

Das Apple Keyboard unterscheidet bei der Tatstatur zwischen der **[ALT]** Taste **Links** und **Rechts** neben der Leertaste (Space). So kann bei Verwendung der rechten **[ALT]** Taste unter WINDOWS auf das zusätzlich Betätigen der **[CTRL]** Taste verzichtet werden.

5. Erweiterte Funktionen (AddOn's)

m..myhome bietet neben der umfangreichen Kommandoschnittstelle über weitere Funktionen wie dem CommandServer zur Einbindung von Fremdprotokollen in den **m..myhome** Befehlssatz oder dem GUI Interface zum erstellen von „Button" Steuerleisten. Diese zusätzlichen Funktionen ermöglichen den Einsatz von **m..myhome** als eigenständige Steuerzentrale.

All diese Schnittstellen sind wie nachfolgend beschrieben offen gelegt und können durch den Nutzer beliebig verändert und auch erweitert werden.

5.1. Die CommandServer Erweiterung

Der CommandServer dient dazu Protokolle zusätzlicher Geräte in den **m..myhome** Befehlssatz zu implementieren. So ist es z.B. notwendig neben dem Steuern des Mac – Systems auch zus. Peripherie wie z.B. ein Multiroom Audiosystem, den angeschlossenen Plasma Screen, einen Beamer, oder einen IR Emitter in die Befehlskette einzubinden. **m..myhome** kann somit auch als komplexe Steuerzentrale eingesetzt werden. Vor allem wird durch diese Technologie dem Anwender die Integration solcher Systeme erheblich vereinfacht, da er sich lediglich mit dem **m..myhome** Befehlssatz auseinander setzen muss. Die für den CommandServer notwendigen Definitionstabellen werden auf unserer Hompage zum freien Download bereit gestellt. Das Angebot an Definitionstabellen wird permanent erweitert.

Die **m..myhome** Grundlizenz (Gate) verfügt über eine CommandServer Lizenz. Dies bedeutet, dass ein Fremdgerät gesteuert werden kann. Die **m..myhome** Gate+ Lizenz verfügt über 10 CommandServer Lizenzen und kann bis auf max. 25 CommandServer Lizenzen ausgebaut werden.

m..myhome prüft beim Start des Services, ob Definitionstabellen verfügbar sind. Sofern entsprechende Einträge gefunden wurden, werden die Befehlsketten eingebunden und stehen der Steuerung zur Verfügung.

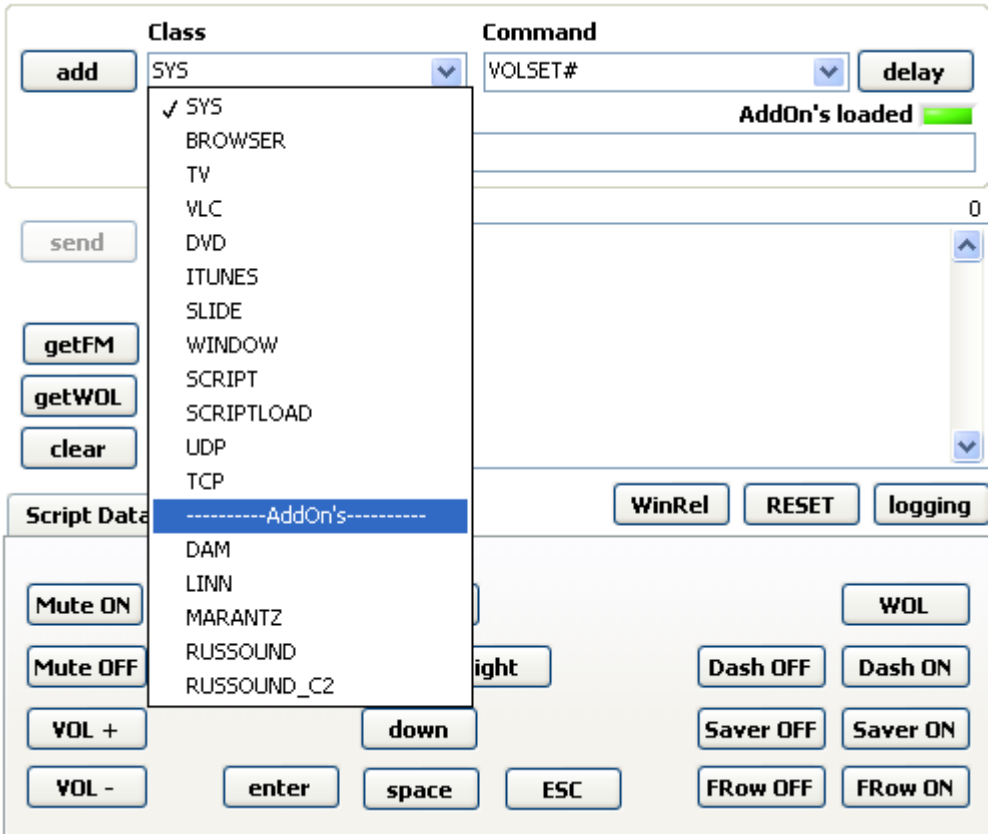
Hinweis:

Bitte achten Sie darauf, dass die Anzahl der Definitionstabellen in dem entsprechenden Verzeichnis die Anzahl der lizenzierten CommandServer nicht überschreitet. **m..myhome** ***implementiert die Tabellen sequentiell. D.h. die Tabellen werden in der Reihenfolge gelesen, wie sie vom Dateisystem aufgrund der Namensgebung zur Verfügung gestellt werden. Ist die Anzahl der Tabellen nun größer als die Anzahl der Lizenzen, werden weitere Tabellen ignoriert.***

Sind entsprechende Definitionstabellen vorhanden, können die Befehle der CommandServer Definitionen über den **m..myhome** scriptingClient abgerufen werden. Öffnen Sie hierfür den Dateidialog und bestätigen Sie „AddOn Commands laden".

TIP:

Benennen Sie die CommandServer Definitionstabellen nach Klassifizierungen wie z.B. BEAMER.csv oder TELEVISION.csv. Benennen Sie vorhandene Tabellen ggf. einfach um. Somit können Sie komplexe Scripte ohne weitere Veränderungen einsetzen. Die Auswahl welche Steuercodes notwendig sind bestimmt allein der Inhalt der Definitionstabellen. Bitte achten Sie bei der Wahl des .csv-Namens darauf, keine bereits existierenden Klassenbezeichnungen (SYS,BROWSER,ITUNES,TV,...) zu verwenden, damit eine eindeutige Zuordnung der Commandserver-Klasse gewährleistet ist.



Nach dem Einlesen der Definitionstabellen erweitern sich die Befehlsklassen um den Inhalt der Definitionstabellen.

5.1.1 Aufbau der CommandServer Definitionstabellen

Die CommandServer Definitionstabellen unterliegen einem einfachen Aufbau, ähnlich einer .ini Datei wie Sie in der WINDOWS Umgebung üblich sind.

Die CommandServer-Definitionen müssen sich im Ordner:

[home] / Dokumente / macmyhome / addons

befinden, der Dateityp ist CSV, der Dateiname (ohne Suffix) entspricht der durch den CommandServer generierten Befehlsklasse im **m..myhome** Protokoll.

Jede CSV-Datei (Trennzeichen „Semikolon") besteht aus maximal 3 Abschnitten, der 1. Abschnitt ist zwingend, die anderen beiden optional:

Der Commandserver arbeitet Befehle parallel ab und kann damit mehrere Geräte gleichzeitig steuern. Die TIMEOUT-Einstellung hat keinen Einfluß darauf, wie schnell neue Befehle abgesetzt werden können.

Wird während eines laufenden Timeouts ein neuer Befehl an das selbe Gerät gesendet, wird der Timeout unterbrochen, der neue Befehl abgesetzt und der Timeout des neuen Befehls neu gestartet.

1. [CONFIG] *Grundkonfiguration, Protokolldefinitionen*

Variable

SERVERIP {*Device-IP}: oder „LOCAL" (bei USB/RS232 Betrieb)*

SERVERPORT *{TCP/UDP Device-Port}, oder bei Verwendung von USB/RS232 Adaptern AUTO/{Serieller Port};{Geschwindigkeit};{Datenbits};{Parität};{Stopbits};{Flußkontrolle}*

CLIENTIP {*Device-IP}: gibt eine weitere IP an, an die das Reply des Commandservers gesendet wird*

CLIENTPORT {*Device-PORT}: gibt den Port der Clientip an, an die das Reply gesendet werden soll. CLIENTIP und CLIENTPORT müssen beide definiert sein*

SERVERMODE; *Übertragungsart: "ASCII" oder "HEX""*

SERVERTIMEOUT; *Zeit (in Sekunden), die auf eine möglich Antwort gewartet wird. "0" = nicht warten, „NONE" = Verbindung dauerhaft halten*

Wenn **SERVERTIMEOUT;NONE** definiert wurde,
wird nach dem Absetzen eines Befehls die Verbindung dauerhaft aufrecht erhalten und eintreffende Daten des Gerätes interpretiert (vgl. Eventserver). Dies Funktioniert mit UDP,TCP und seriellen Verbindungen. Der Timeout lässt sich natürlich auch weiterhin kommandobezogen definieren

SERVERTIMEOUT_IN_MS;
Zeit (in ms), die auf eine möglich Antwort gewartet wird. "0" = nicht warten

Wenn SERVERTIMEOUT und SERVERTIMEOUT_IN_MS gleichzeitig definiert sind, geht SERVER-TIMEOUT_IN_MS vor.

SERVERPROTOCOL; *Protokoll: "UDP", "TCP", "HTTP" oder „SERIAL" für USB/RS232 Adapter*

SERVERTYPE; *Reserviert für fest implementierte Protokolle*

MATCHING;NORMAL *Einstellung der Abhandlung für das Parsen der MAPPINGS „NORMAL" (Default) oder „FULL"*

CMDPREFIX; *fügt vor jedem Command Eintrag entsprechend Zeichen ein*

CMDSUFFIX; *fügt jedem Command Eintrag entsprechend Zeichen an*

MAPPREFIX; *fügt vor jedem Mappings-Eintrag entsprechend Zeichen ein (analog zu CMDPREFIX)*

MAPSUFFIX; *fügt jedem Mappings-Eintrag entsprechend Zeichen an (analog zu CMDSUFFIX)*

OPTIONAL (nur bei Verwendung SERVERPROTOCOL=HTTP)

USERNAME;	*HTTP Anmeldung Username*
PASSWORD;	*HTTP Anmeldung Password*

TIP:

CMDxxx Parameter oder MAPxxx sind optional und werden nur berücksichtigt, wenn kein SERVERTYPE definiert wurde. Verwenden Sie diese Option, wenn das verwendete Protokoll wiederkehrende Start- oder Endzeichen beinhaltet.

CMDPREFIX; Hi there,
CMDSUFFIX; Have a nice time.

[COMMANDS];
MYSELF;my name is Mike.

erzeugt folgenden String:
Hi there, my name is Mike. Have a nice time.

---

2. [COMMANDS] Kommando-Definitionen

{CMD};{Sequenz}

weist einem frei definierbaren Kommando {CMD} eine zu versendende Sequenz {Sequenz} zu, die im entsprechenden Format (ASCII/HEX) angegeben wird. In {Sequenz} können folgende Spezialzeichen vorkommen:

"\#"	*fügt das mit {CMD} übergebene Argument an der entsprechenden Stelle in {Sequenz} ein.*

Im ASCII-Modus wird das Argument ohne weitere Wandlung eingefügt; Im HEX-Modus wird das Argument vorzeichenbehaftet (1 Byte) umgewandelt und anschließend eingefügt. Mehrere Argumente können durch Kommata getrennt übergeben werden.

Die Sequenz wird in der Reihenfolge der übergebenen Kommandos aufgebaut und nach Terminierung der Klasse abgesendet.

Beispiele:

a) SERVERMODE;ASCII

SWITCH;SCHALTE_AUF_DEN_\#-KANAL
RECORD;_AND_RECORD_\#_AND_\#

Die Befehle:

<[CLASS]><SWITCH=ZDF><RECORD=ARD,RTL><[/CLASS]>

erzeugen folgende Sequenz:

SCHALTE_AUF_DEN_ZDF-KANAL_AND_RECORD_ARD_AND_RTL

Dem ASCII String können auch nicht darstellbare Zeichen übergeben werden. Diese Zeichen werden ebenso eingebunden wie ein Argument. Bitte beachten Sie, dass jeweils nur ein Zeichen übergeben werden kann. Diese Zeichen werden wie folgt übergeben:

SWITCH;SCHALTE_AUF_DEN_\#-KANAL\0x0D

\0x ist hierbei der Platzhalter für folgende HEX Bytes, **0D** ist das zu sendende Byte (in diesem Fall ein Zeilenvorschub bzw. ein RETURN). Bitte beachten Sie, dass je Byte der Platzhalter zu verwenden ist. Also **\0x0D\0x0C\0xFF** … usw.

b) SERVERMODE;HEX

SWITCH;07AB\#04
RECORD;07CF\#BD\#06

Die Befehle:

<[CLASS]><SWITCH=1><RECORD=-1,2></[CLASS]>

erzeugen folgende Sequenz:

07AB010407CFFFBD0206

c) SERVERMODE;HTTP

PLAY;/cgi-bin/videocontrol?command=play
STOP;/cgi-bin/videocontrol?command=stop

Die Befehle:

<[CLASS]><PLAY></[CLASS]>

erzeugen folgende Sequenz:

/cgi-bin/videocontrol?command=play

Der Servermode HTTP dient z.B. zur Ansteuerung von Geräten, die über eine CGI Schnittstelle bzw. ein WEB Interface verfügen. Ebenso können hierüber Webabragen generiert werden.

TIP:

Zusätzlich kann für jedes Kommando der Timeout auch separat gesetzt werden:

Beispiel:

PLAY;$PLAY;800	*- Timeout 800ms*
PAUSE;$PAUSE;1700	*- Timeout 1,7 s*
STOP;$STOP	*- globaler Timeout, wie über SERVERTIMEOUT oder SERVERTIMEOUT_IN_MS*

Hinweis:

Werden mehrere Befehle in derselben Sequenz abgesetzt, gilt der Timeout des jeweils letzten Befehls:

Beispiel:

<LINN><PLAY><PAUSE></LINN>	*- Timeout: 1,7s*
<LINN><PAUSE><PLAY></LINN>	*- Timeout: 800ms*
<LINN><PAUSE><PLAY><STOP></LINN>	*. Timeout: globale Einstellung*

3. [MAPPINGS] (optional) *Interpretation der Rückgabesequenz*

{Map String};{mmh-Bezeichnung};{Action};ONCHANGE;{Bedingung}

{Map String}	*Eingangsstring, der unter Verwendung der Parser Optionen ausgewertet Werden kann. Bei Verwendung ohne Parser ist die Bedingung erfüllt, wenn Eingangsstring = {Map String}. Bei Verwendung mit Parsern ist die Bedingung erfüllt, wenn ein erfolgreiches Match erfolgt. Der Inhalt der Parser Option wird als Argument übergeben.*
{mmh-Bezeichnung}	*Bezeichner für das Ergebnis des erfolgten Matches. Das Argument wird automatisch angefügt.*
{Action}	***m..myhome** Befehlskette und oder Script, welches bei erfolgtem Match ausgeführt werden soll.*
ONCHANGE	*Die Option ONCHANGE wird verwendet, wenn der Wert nur bei Wertänderung gesendet werden soll. Bleibt das Feld leer, wird der Wert immer gesendet.*
{Bedingung}	*Wenn definiert, wird vorherige Action nur ausgeführt, wenn Bedingung erfüllt. Es können Komperatoren („**>**“,“**<**“,“**=**“) eingesetzt werden. Auch Kombinationen wie zB. „**<>**" (ungleich) oder „**>=**" (kleiner oder gleich) können definiert werden.*

Durchsucht den Eingangsstring **{Map-String}** gem. den nachfolgenden Bedingungen und übergibt die gefundene Zeichenkette (Match) als Argument an das Feld **{mmh-Bezeichnung}**. Hierbei kann in **{mmh-Bezeichnung}** auch ein frei definierbaren String als Klartext Bezeichnung eingetragen werden (s. Beispiel). In **{Action}** wird die **m..myhome** Befehlskette und/oder das Script eingetragen, welches zur Ausführung kommt, wenn **{Bedingung}** erfüllt. Folgende Spezialzeichen können in **{Map String}** als Option für das Matchen oder Parsen verwendet werden:

Parser Optionen

"\^"	*Matcht genau ein Zeichen und fügt es dem Argument an.*
"\?"	*Match´t genau ein Zeichen und ignoriert es (Wildcard)*
"\^" und "\?"	*dürfen an beliebiger Stelle stehen*
"*"	*Wildcard-Match und fügt es dem Argument an*
"\%"	*ignoriert beliebig viele Zeichen*

Weitere Regeln:

Kombinationen von **"\^"** und **"\?"** sind in beliebiger Länge möglich. Also Definitionen wie zB **„\^^^^“** für 4 Zeichen

Wird **„\^“** und/oder **„*“** kombiniert, wird das Argument entsprechend um die jeweiligen Zeichen ergänzt.

die Einträge im MAPPINGS-Abschnitt werden gem. dem Schalter **MATCHING=NORMAL** (Reststring nach Match) oder

MATCHING=FULL (gesamter Eingangsstring), zeilenweise abgearbeitet.

Bei einem Match (**MATCHING=NORMAL)** werden weitere Einträge ignoriert und die verbliebenen Zeichen in der Rückgabesequenz erneut evaluiert. Daher ist es sinnvoll, komplexe/lange Match-Strings an den Anfang des Mapping-Bereichs zu verschieben, um eine saubere Interpretation sicherzustellen. Im ASCII-Modus wird das Argument ohne weitere Wandlung angefügt; Im HEX-Modus wird das Argument in Dezimal umgewandelt und anschließend angefügt. Es können auch Hex-Werte (z.B. "\0x0D", genau wie im COMMANDS-Bereich) vorkommen, auf die gematcht werden kann.

Beispiel (SERVERMODE;HEX):

07AA\##??;VOLUME
07EF\##04;CHANNEL

Die Rückgabesequenz:

07EF040407AA1008

erzeugt die folgende Klartext-Sequenz:

<[CLASS]><CHANNEL=4><VOLUME=16></[CLASS]>

Beispiel (SERVERMODE;ASCII):

TITLE=*|;TITLE

Die Rückgabesequenz:

<ITUNES>TITLE=When the dog bells|ALBUM

erzeugt die folgende Klartext-Sequenz:

<[CLASS]><TITLE=When the dog bells></[CLASS]>

Beispiel (SERVERMODE;ASCII) Erweitert:

<SYS>\%VOL=*|;VOLUME;<SYS><ENABLESS></SYS>;;<=5

Die Rückgabesequenz:

<SYS>GETVOL=EXT|VOL=4|...</SYS>

erzeugt die folgende Klartext-Sequenz:

<[CLASS]><VOLUME=4></[CLASS]>

und löst gleichzeitig den Bildschirmschoner aus, da das Argument <=5

Beispiel (SERVERMODE;HTTP):

<head><title>standby</title></head><body>*</body></html>;STANDBY

Die Rückgabesequenz:

<head><title>standby</title></head><body>on</body></html>

erzeugt die folgende Klartext-Sequenz:

<[CLASS]>STANDBY=ON</[CLASS]>

Hinweis:

sobald eine einzige Mappings-Zeile existiert, wird nicht evaluierbarer Inhalt der Rückgabesequenz unterdrückt. So ist es möglich, nur die relevanten Daten aus einer Rückgabesequenz zu extrahieren, ohne Definitionen für alle möglichen Kombinationen anlegen zu müssen. Existiert kein MAPPINGS-Abschnitt, wird der raw-Output ausgegeben.

Die Kommunikation lässt sich im raw-Format (also nach der Umwandlung bzw. vor der Rückwandlung) jederzeit über den Logging-Monitor im **m..myhome** scripting Client überwachen.

Für das weitere Verständnis erklären wir anhand einer einfachen Definitionstabelle den Aufbau einer Tabelle für einen LINN Verstärker.

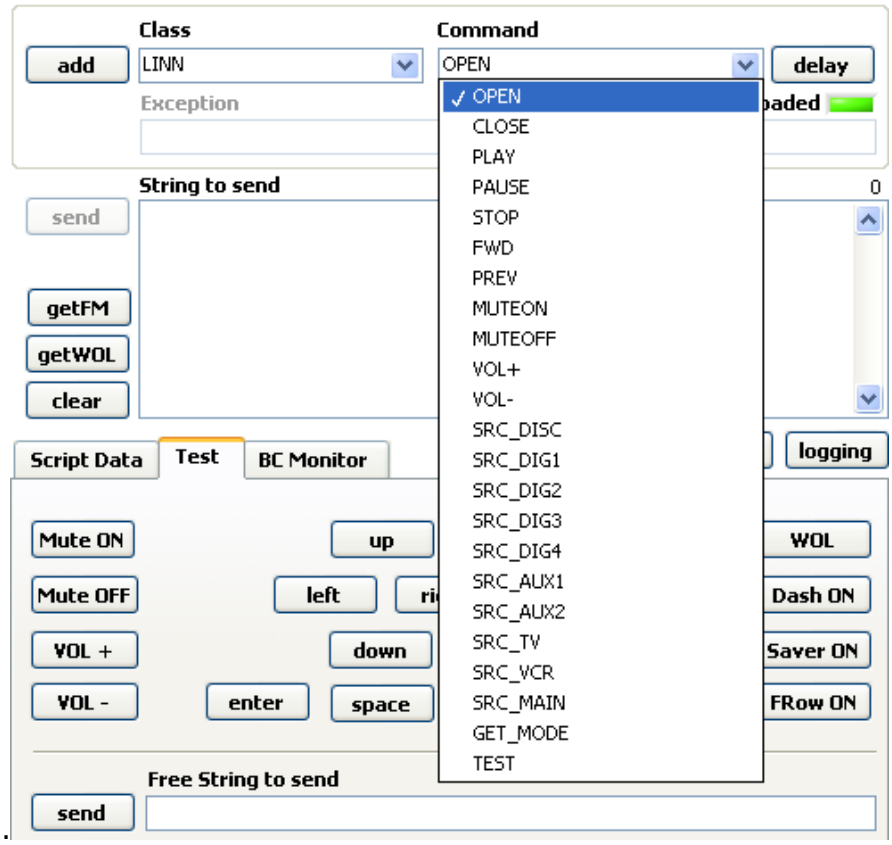
Linn.csv

[CONFIG]	
SERVERIP	192.168.50.201
SERVERPORT	4001
SERVERMODE	ASCII
SERVERTIMEOUT	0
[COMMANDS]	
OPEN	$OPEN$\x0d\x0a
CLOSE	$CLOSE$\x0d\x0a
PLAY	$PLAY$\x0d\x0a
PAUSE	$PAUSE$\x0d\x0a
STOP	$STOP$\x0d\x0a
FWD	$SKIP +$\x0d\x0a
PREV	$SKIP -$\x0d\x0a
MUTEON	$MUTE\x20ON$\x0d\x0a
MUTEOFF	$MUTE\x20OFF$\x0d\x0a
VOL+	$VOLUME\x20+$\x0d\x0a
VOL-	$VOLUME\x20-$\x0d\x0a
SRC_DISC	$LISTEN\x20DISC$\x0d\x0a
SRC_DIG1	$LISTEN\x20DIG1$\x0d\x0a
SRC_AUX1	$LISTEN\x20AUX1$\x0d\x0a
SRC_AUX2	$LISTEN\x20AUX2$\x0d\x0a
SRC_TV	$LISTEN\x20TV$\x0d\x0a
SRC_VCR	$LISTEN\x20VCR$\x0d\x0a
SRC_MAIN	$LISTEN\x20MAIN$\x0d\x0a
GET_MODE	$MODE$\x0d\x0a

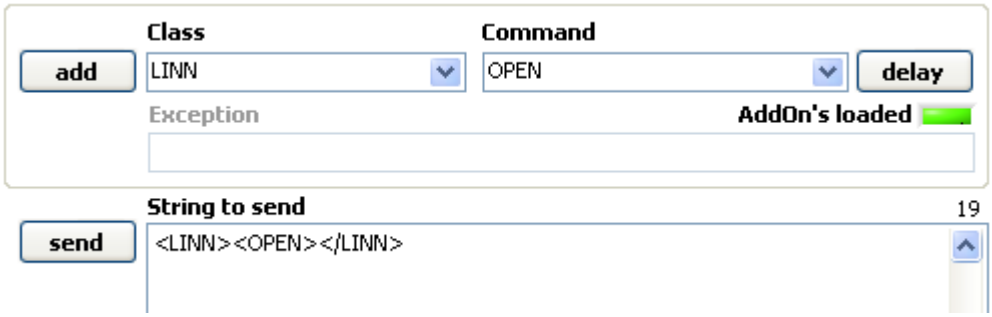
Wie vor beschrieben befindet sich im Abschnitt [CONFIG] die Einstellung für die Konfiguration der Kommunikationsparameter des zu steuernden Device. Da der LINN Unidisc über keine Netzwerkschnittstelle verfügt wurde hier ein MOXA LAN/RS232 Adapter eingesetzt. Der Moxa wurde mit der IP Adresse 192.168.50.201 belegt und der Port auf 4001 eingestellt. Das LINN Protokoll ist ein reines ASCII Protokoll. Ein Feedback wird nicht ausgewertet (TIMEOUT=0). Das Fehlen der Einstellungen SERVERPROTOCOL hat zur Folge, das dass UDP Protokoll (Standard) eingestellt wurde. Da dieses Protokoll nicht intern weiterverarbeitet wird, fehlt auch der Eintrag SERVERTYPE.

Unter dem Bereich [COMMANDS] sind nun die eigentlichen Befehle eingetragen. Gem. Herstellerdokumentation muss dem LINN folgende Befehlskette zum Öffnen des Laufwerkschachtes übertragen werden „$OPEN$\x0d\x0a". Dieser Befehl soll zukünftig im **m..myhome** Protokoll unter<OPEN> zur Verfügung gestellt werden. Ebenso wird mit allen weiteren notwendigen Befehlen verfahren.

Ist die Tabelle erstellt und in dem entsprechenden Verzeichnis auf Ihrem Mac System abgelegt, muss der **m..myhome** Service neu gestartet werden bzw. ein „RESET" über den **m..myhome** scripting Client ausgelöst werden. Danach führen Sie im Dateidialog den Befehl „AddOn Commands laden" aus. Wählen Sie nun die Klasse „LINN", die aufgrund des Dateinamens der Definitionstabelle definiert wurde, aus und betrachten Sie die neuen Befehle



Sie sehen, dass nun alle Befehle, die Bestandteil der CommandServer Definitionstabellen sind, zur Verfügung gestellt werden. Wählen Sie nun den „open" aus und fügen Sie diesen in die „String to send" Maske ein.



Senden Sie den Befehl an Ihr Mac System und beobachten Sie dabei den Eintrag in dem Logging Monitor.

Sie sehen, dass **m..myhome** die Daten entsprechend den Einstellungen der Definitionstabelle behandelt und das Notwendige Datenpaket an die eingestellte Zieladresse sendet.

5.1.2 Reservierte (extended) Servertypen [SERVERTYPE]

Wie vor beschrieben, gibt es Serverklassen, die bereits fest in **m..myhome** implementiert wurden. Diese speziellen Servertypen können nur vom **m..myhome** Entwicklerteam eingebunden werden und dienen der Vereinfachung in der Verarbeitung komplexer Protokollstrukturen. Hierüber ist es beispielsweise möglich, Checksummen Berechnungen o.ä. zu hinterlegen. Generell werden derartige Protokolle komplett vom **m..myhome** Entwicklerteam eingebunden und die entsprechenden Definitionstabellen, sofern vorhanden, frei zur Verfügung gestellt.

Selbstverständlich können entsprechende Protokolle auch über die vor beschriebenen möglichen Verfahren eingebunden werden, jedoch kann bei Protokollen mit z.B. anhängiger Checksumme kein dynamischer Wert übergeben werden. Dies würde z.B. bei einem Multiroom System bedeuten, dass je Befehl eine Zeile in der Definitionstabelle eingefügt werden muss.

Zurzeit sind folgende extended Servertypen eingebunden:

- RUS	*ermittelt und sendet die Checksumme für das Russound Protokoll (HEX)*
- ADNOTAM	*ermittelt und sendet die Checksumme für das Adnotam Protokoll (HEX)*

5.1.3 Verwendung von USB/RS232 Umsetzern

Ab der **m..myhome** Version 1.0.45 wird die Verwendung der Support lokaler USB/RS232 Adapter/Umsetzer supportet. Hierbei kann auf teuere IP/RS232 Umsetzer verzichtet werden, sofern Ihre Installation dies zulässt. Nachfolgend wird die Vorgehensweise zur Verwendung dieser Adapter beschrieben bzw die notwendigen Eintellungen in der csv Datei verdeutlicht.

Variable

SERVERIP;LOCAL	*Betriebsart für lokale Devices*
SERVERPORT	*AUTO/{Serieller Port};* *{Geschwindigkeit};{Datenbits};* *{Parität};{Stopbits};* *{Flußkontrolle}*

Erklärung der Variablen SERVERPORT bei SERVERIP=LOCAL

Bei Verwendung von USB/RS232 Adapter muss die Einstellung SERVERIP=LOCAL erfolgen.

AUTO	*Sucht den 1. seriellen Port im System und nutzt ihn automatisch. Die Nutzung macht eigentlich nur dann Sinn, wenn man nur einen USB/RS232 Adapter installiert hat.*
{Serieller Port}	*der Schnittstellenname im Format /dev/cu.xxxxxxx*

Die Verwendung des Befehls GETSERIALPORTLIST aus der SYS Klasse gibt Auskunft über die vom System verwendeten Schnittstellen- bzw. Devicenamen.

{Geschwindigkeit}	*die Geschwindigkeit, mit der der serielle Port betrieben werden soll, getestet wurden 9600,19200 und 115200 bps*
{Datenbits}	*Anzahl der Datenbits (5-8)*
{Parität}	*'n' -> keine* *'e' -> gerade* *'o' -> ungerade*
{Stopbits}	*Anzahl der Stopbits (1-2)*
{Flusskontrolle}	*‚NONE', ‚RTSCTS', ‚DTRDSR', ‚XONXOFF'*,

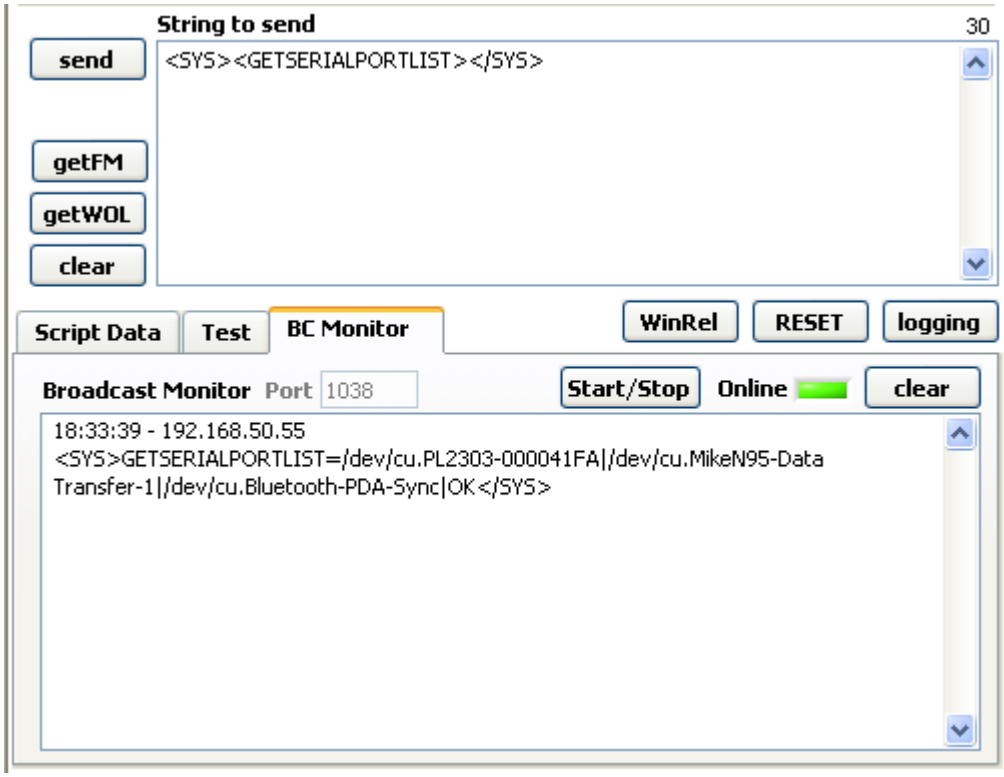
Beispiel Einstellung für USB/RS232 Adapter

SERVERIP;LOCAL
SERVERPORT;AUTO;115200;8;n;1;RTSCTS
SERVERPROTOCOL;SERIAL

Sucht und öffnet den 1. gefundenen lokalen USB/RS232 Adapter und stellte eine COM Schnittstelle mit den Parametern 115200,8,n,1 und aktivierter RTSCTS Flusskontrolle bereit.

Um Auskunft über die teilweise doch kryptischen Device Namen zu erhalten, wurde der Befehl **GETSERIALPORTLIST** aus der Klasse der Systembefehle eingebunden.

Verwenden Sie einfach den Scripting Client um an die entsprechenden Namen zu gelangen.



Alle Rückmeldungen mit dem Inhalt:

<SYS>GETSERIALPORTLIST=**/dev/cu.PL2303-**0000201A|

sind entsprechende USB/RS232 Device Treiber. In gezeigtem Fall wäre der korrekte Eintrag in der csv Datei:

SERVERPORT;/dev/cu.PL2303-0000201A;115200;8;n;1;RTSCTS

Entsprechend zu verwenden, wenn ein Device unter seiner direkten Adresse angesprochen werden soll bzw. wenn mehr als ein USB/RS232 Device an dem System verwendet wird.

Hinweis:

Die erste in der zurück gelieferten Liste auftauchende Bezeichnung entspricht der Auto-Funktion in der .csv

TIP:

Zum Einrichten eines Systems ist es sinnvoll, nach und nach die Konverter einzustöpseln und zwischenzeitlich immer wieder den o.g. Befehl aufzurufen, um die richtige Zuordnung und den entsprechenden Portnamen zu bekommen. Die Portbezeichnung richtet sich danach, in welchen USB-Port der Konverter gesteckt wird. Das kann natürlich auch an einem externen USB- Hub erfolgen.

Wir haben Adapter mit dem *Profilic PL- 2303 Chipsatz* erfolgreich testen können und empfehlen dementsprechend auch ausschließlich deren Verwendung.

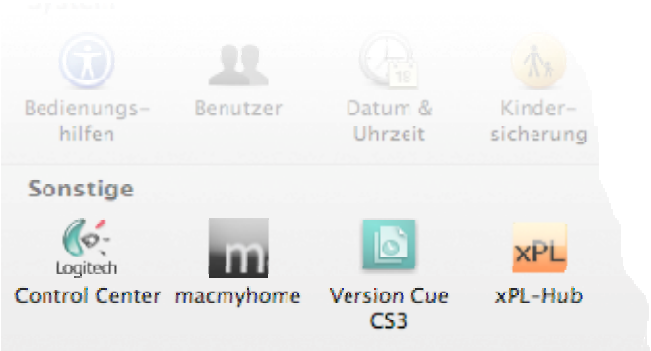
Den entsprechenden Mac Treiberdownload für den Prolific PL2303 (der Chipsatz wird von zig verschiedenen Herstellern verwendet)finden Sie unter nachfolgender Adresse. Ebenso befindet sich der Treiber auf der beigefügten CD unter dem Verzeichnis **\Tools und Treiber\Profilic**.

http://sourceforge.net/projects/osx-pl2303/

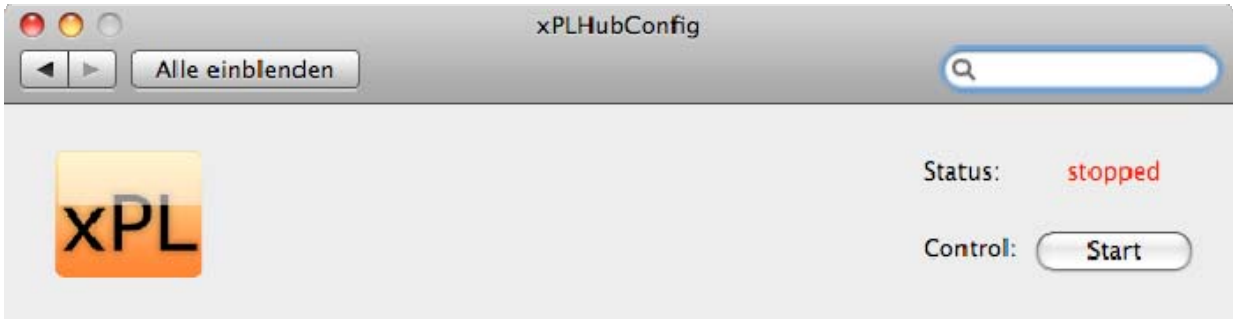
Bitte beachten Sie, dass die Verwendung des Treibers der *GNU General Public License (GPL*) unterliegt.

5.2. Slimserver, xPL

m..myhome unterstützt zur Ansteuerung der Squeezeboxen/Slimdevice der Fa.Logitech das xPL Protokoll. xPL steht für eXtremly simPle protocoL. Weitere Informationen zu xPL finden Sie unter *http://wiki.xplproject.org.uk*. Um xPL auf Ihren Mac einsetzen zu können, muss der xPL-Hub auf Ihrem System installiert werden. Sie finden die Software (xPL Hub.mpkg) in dem Downloadbereich auf unserer Hompage oder in dem Ordner \m..myhome\m..myhomeAppleOSXInstaller der m..myhome CD. Installieren Sie das Package und folgen Sie den Anweisungen. Bitte beachten Sie, dass der xPL-Service gestartet wird. Sie finden den xPL Service ebenso wie den m..myhome Service in den Systemeinstellungen.



Öffnen Sie die xPL Hub Konfiguration durch einen Mouseklick auf dem xPL Icon.



In dem Dialogfenster kann der xPL Service entsprechend gestartet oder gestoppt werden. Diese Einstellung bleibt auch nach einem Neustart erhalten.

Das Panel zeigt den Status des Hubs in Echtzeit an. Der Hub läuft völlig unabhängig von **m..myhome**. Aufgrund der xPL-Spezifikation kann es bei xPL-Devices, die vor dem Hub gestartet wurden, bis zu 5 Minuten dauern, bis sie sich beim Hub angemeldet haben.

Um die xPL Dienste zur Ansteuerung der Squeeze Box nutzen zu können, muss die xPL-Schnittstelle im Squeeze Center freigegeben werden. Entsprechende Auswahl finden Sie unter Plugins in den Squeeze Center Einstellungen. Sofern Sie über keine Squeeze Box verfügen, können Sie sich auch die SoftSqueeze verwenden. Entsprechenden Link zum Installieren der Software finden Sie in den Squeeze Center Einstellungen unter Players.

Hinweise über die Squeeze Box und dem Squeeze Center finden Sie unter *http://www.logitech.com* oder unter *http://www.slimdevices.com*

5.2.1. Konfiguration der xPL Klasse

Um die xPL Klasse unter **m..myhome** nutzen zu können, müssen entsprechende Konfigurations Dateien unter [user]\Dokumente\m..myhome\xpl vorhanden sein. Das Verzeichnis wird mit der Installation des **m..myhome** Services automatisch angelegt. Der Aufbau der Konfigurationstabellen erfolgt konform dem Aufbau der CommandServer Definitionen. Je Squeezebox ist entsprechend eine Tabelle anzulegen. Der Name der Tabelle definiert die Klasse (z.B. SqueezeSchlafen.csv, SqueezeBad.csv).

In der Tabelle muss lediglich die Adresse der Squeeze Box angepasst werden, die über **m..myhome** angesprochen werden soll. Die Einstellung finden Sie unter dem Bereich [CONFIG] unter TARGET.

```
[CONFIG]
TARGET      Slimdev-slimserv.Squeezebox
```

Ändern Sie den Namen „**Squeezebox**“ (Standard) entsprechend in den verwendeten Namen um. Dies gilt selbstverständlich für jede verwendete Tabelle. Bitte beachten Sie, dass Änderungen an den Tabellen erst nach einem Restart/Reset des **m..myhome** Services übernommen werden.

Der Befehlssatz kann bei Bedarf entsprechend erweitert werden. Hierzu sind jedoch Kenntnisse im Umgang mit dem xPL Protokoll notwendig. Für Anwender, die sich mit dem xPL Protokoll beschäftigen ist nachfolgend der Aufbau der Definitionsstruktur erläutert. Dieser Abschnitt kann bei Standard Anwendungen übersprungen werden.

Erklärung der .csv-Struktur (für Experten):

Bereich [CONFIG]:

TARGET	*gibt das xPL-Target an, an das die Kommandos versendet werden sollen, "*" als Wildcard ist erlaubt. Der Name muss den xPL-Spezifikationen entsprechen.*
Bereich [SCHEMA={Profile}]	*definiert das xPL-Befehlsschema, für das die nachfolgenden Kommandos definiert sind.* ***Es werden alle xPL Schemata unterstützt.***
{CMD};{XPL_KEY=XPL_VAL}	*weist einem beliebigen Kommando das entsprechende xPL-Key-Value-Pärchen zu. Argumente können entsprechend der CommandServer-Syntax mit übergeben werden (siehe slimserver.csv)*
Bereich [WATCH={Profile}]	*definiert das xPL-Schema, das bei xPL- Status Meldungen evaluiert werden soll.*
{XPLKEY};{NAME}	*übersetzt den empfangenen XPLKEY in einen frei definierbares Schlüsselwort. das Argument des empfangenen XPLKEY's wird an dieses Schlüsselwort angehängt.*

XPLKEY's, für die kein NAME definiert wurde, werden in der Broadcast-Ausgabe ignoriert.
m..myhome hat die xPL-Adresse [vendorID-deviceID.instanceID] „ibs-mmh.{Mac Seriennummer}", also z.b. ibs-mmh.YM8031UVX89

Ausführliche Protokoll Definitionen finden Sie hier:
http://wiki.xplproject.org.uk/index.php/Schema_-_AUDIO.SLIMSERV

Nachdem nun alles korrekt eingerichtet wurde, können Sie über den ScriptingClient die Kommunikation testen. Die neuen Befehlsdefinitionen können abgerufen werden, indem Sie wie vor beschrieben die AddOn Befehle laden [Datei], [AddOn Commands laden]. Die Slimserver Befehlsdefinitionen stehen nun unter der entsprechenden Klasse (Name der .csv Datei) bereit.

Leerzeichen in Playlisten müssen durch ein **%20** ersetzt werden.

Folgendes Beispiel spielt die Playliste (Wiedergabeliste) **test test**

<SLIMSERVER_MMH><ADDANDPLAY=test%20test></SLIMSERVER_MMH>

Durch die Verwendung von *Formatoptionen* kann beim Aufruf das Ersetzen der Blancks in die HTML konforme Ausgabe „%20" erzwungen werden.

<SLIMSERVER_MMH><ADDANDPLAY=test test{%r ,%20}**></SLIMSERVER_MMH>**

wandelt den Befehl in <ADDANDPLAY =test%20test>. Die Umwandlung erfolgt hierbei für jedes folgende „Blanck"
Im Broadcast-Monitor werden, wie bei iTunes auch, Titelinformationen etc. übertragen. Die genaue Erklärung der Befehle/Commands sind in der XPL CommandServer Beschreibung die Sie mit dem xPL AddOn bei der Registrierung erhalten.

Um xPL verwenden zu können, benötigen Sie mindesten eine **m..myhome Gate+** Lizenz, sowie das xPL AddOn. Das AddOn ist auf max. 25 Definitionstabellen begrenzt.

Die Schnittstelle sowie der Befehlsatz werden in den nächsten Versionen noch erweitert.

5.3. Event Server

Der Eventserver ermöglicht **m..myhome** Kontrollsequenzen fremder Systeme/Protokolle direkt zu verarbeiten, auszuwerten und entsprechend **m..myhome** Scripte anzutriggern.

Dies wird beispielsweise benötigt, wenn auf ein „Feedback" eines beliebigen Gerätes (z.B. Einschalten des Beimers über die Beamer eigene Fernbedienung) reagiert werden soll. So ist es beispielsweise möglich folgende Funktionalität zu realisieren…

… IF BEAMER = ON, send SCRIPT PRÄSENTATION

Eventdefinitionen werden im Unterordner **[user]\dokumente\m..myhome\events** abgelegt. Das Dateiformat ist ebenfalls *.csv. Der [Dateiname].csv kann frei gewählt werden. Bitte beachten Sie, dass maximal 25 Event Definitionstabellen möglich sind. Ebenso wird für das generelle Verarbeiten der Event Definitionen eine gesonderte Lizenz benötigt.

Beispiel einer Event Definition:

Ein externes Gerät sendet folgende ASCII Codes über das Netzwerk:

[CONFIG]; **ACTIVE;YES**	*„YES" (default) oder „NO", aktiviert/deaktiviert den Eventserver*
DEBUG;NO	*„Yes" oder „NO" (default), aktiviert/deaktiviert erweiterte LOG Ausgaben*
TRIGGERIP;192.168.1.200	*entfernte IP (Auslöser) oder INTERNAL*
TRIGGERPORT;6200	*lokaler Port (bei Angabe einer IP-Adresse als TRIGGERIP, an dem auf UDP-Datagramme gewartet wird, oder IN/OUT/LOG/BROADCAST (nur in Verbindung mir TRIGGERIP=INTERNAL*
TRIGGERMODE;ASCII	*ASCII oder HEX, wie vom CommandServer bekannt.*
MATCHING;NORMAL	*Einstellung der Abhandlung für das Parsen der TRIGGERS „NORMAL" (Default) oder „FULL"*

Interner Event-Support: gibt man in der .csv für **TRIGGERIP** den WERT "**INTERNAL**" an und für **PORT** "**IN**" bzw. "**OUT**" oder „**LOG**", so wird auf die eingehenden bzw. ausgehenden Kommandostrings gematcht. Dabei spielt es keine Rolle, wie die Kommandos in den Daemon eingespeist werden. **TRIGGERMODE** sollte auf "ASCII" stehen. Der Match ist case-sensitive. Wird für **PORT "BROADCAST"** angegeben, wird auf sämtliche Ausgaben gematcht, die **m..myhome** über den Broadcast-Port ausgibt.

Zur Verdeutlichung (Verwendung der m..myhome Standard Ports vorausgesetzt):

TRIGGERIP;INTERNAL und TRIGGERPORT;OUT
Hat zur Folge, dass die TRIGGERS auf die Antwortsequenzen angewendet werden, die m..myhome auf Port 1037 ausgibt.

TRIGGERIP;INTERNAL und TRIGGERPORT;IN
Hat zur Folge, dass die TRIGGERS auf die Befehlssequenzen angewendet werden, die m..myhome auf Port 1037 geschrieben werden.

TRIGGERIP;INTERNAL und TRIGGERPORT;LOG
Hat zur Folge, dass die TRIGGERS auf die Antwortsequenzen angewendet werden, die m..myhome auf Port 1039 ausgibt.

TRIGGERIP;INTERNAL und TRIGGERPORT;BROADCAST
Hat zur Folge, dass die TRIGGERS auf die Antwortsequenzen angewendet werden, die m..myhome auf Port 1038 bzw. dem Statusport ausgibt. Es werden hierbei nur die Broadcast Ausgaben berücksichtigt, die von dem lokalen **m..myhome** Deamon ausgegeben werden.

[TRIGGERS];
{Match-String};{Action oder Scriptname};{Optionen};{Match-Bedingung}

hello \#;hello.myh;ONCHANGE

Im Abschnitt TRIGGERS wird in der linken Spalte der ASCII- oder HEX-Match angegeben, in der zweiten Spalte folgt das bei einem Match auszuführende Script oder die auszuführende Kommandosequenz. Der Begriff ONCHANGE ist optional und führt das Script nur dann aus, wenn sich das Argument seit dem letzten Trigger geändert hat. Das macht natürlich nur dann Sinn, wenn auch ein Argument evaluiert werden soll. Der Wert, der durch die nachfolgend aufgeführten Parameter extrahiert wird, wird automatisch als „Argument" an das aufgerufene Script übergeben. Das anhängige Script wird ausgeführt, wenn Die Match Bedingung erfüllt ist.

Parser Optionen:

\^	*Matcht ein Zeichen. Je „^" wird jeweils ein Zeichen erwartet.*
*****	*Matcht eine Zeichenkette*
\?	*ignoriert ein Zeichen*
\%	*ignoriert beliebig viele Zeichen*

Wird beispielsweise der empfangene Wert {**R0012?**} mit dem Trigger **R0\^^^?** belegt wäre das Ergebnis **012**. Bei einem Trigger **R*?** Wäre das Ergebnis **0012.**

Bei der Einstellung MATCHING=NORMAL verhält sich der Eventserver wie vor beschrieben. Hierbei ist auch darauf zu achten, dass die Abarbeitung sequentiell erfolgt und der Eingangsstring bei erfolgreichem Match entsprechend gekürzt wird (subtraktives Verhalten). Ein weiterer Trigger innerhalb der csv Datei würde für die Matchbedingung nur noch der restliche Eingangsstring zur Verfügung stehen.

Die Einstellung „**MATCHING=FULL**" ermöglicht ein zeilenweites Parsen des Eingangsstrings. Der Eingangsstring steht also in voller Länge jedem Trigger Eintrag zur Verfügung

Es kann auch auf vergleichende Bedingungen getriggert werden. Hiefür ist das Format der Eventserver Zeile entsprechend erweitert worden.

{Match-String};{Action oder Scriptname};{Optionen};{Bedingung}

Unter {Bedingung} können optional Komperatoren („>","<","=") eingesetzt werden. Auch Kombinationen wie zB. „<>" (ungleich) oder „>=" (kleiner oder gleich) können definiert werden.

Beispiele:

TRIGGERIP;INTERNAL und TRIGGERPORT;BROADCAST

|VOL=*|;<SYS><SETVOL=69></SYS>;ONCHANGE;>=70

Matcht auf die Systemlautstärke, die am Statusport ausgegeben wird. Ist der Wert größer oder gleich 70 wird über einen entsprechenden Befehl die Lautstärke auf „69" zurückgesetzt. Der Vergleich erfolg t nur bei einer Wertänderung (ONCHANGE)

TRIGGERIP;INTERNAL und TRIGGERPORT;IN

<WINDOW>\%<SELECT=3>\%<CREATE>;<WINDOW><SELECT=3><DELAY=3><RELEASE></WINDOWS>;

Matcht auf eine eingehende Befehlssequenz, auf die Befehle **<SELECT=3>** und **<CREATE>** der **<WINDOW>** Klasse. Ist diese Bedingung erfüllt, wird das Fenster nach 3 Sekunden geschlossen.

TRIGGERIP;INTERNAL und TRIGGERPORT;OUT

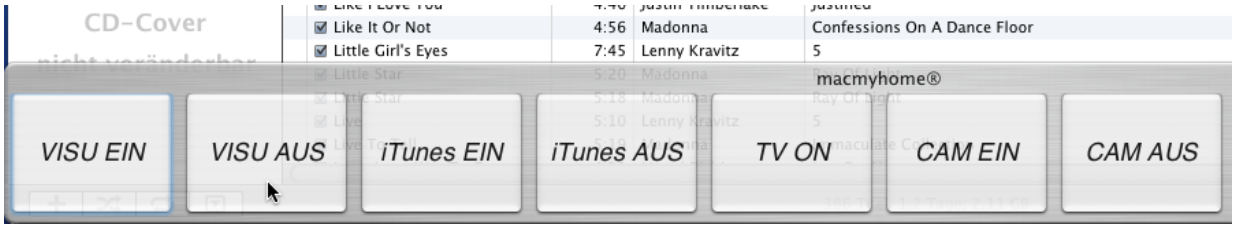
<TV>\%FSON=|OK;<SCRIPT><RUN=MuteAll.myh></SCRIPT>;

Matcht auf die Antwort eines ausgeführten **<TV>..<FSON>** Befehls und führt bei erfolgreichem Match das Script **MuteAll.myh** aus.

Hinweis:
Es sind auch mehrere .Definitonsdateien pro Port möglich. Bei einem gültigen Event wird die Zeilennummer des Matches und der Dateiname der .csv im Logging ausgegeben.

5.4. Die GUI Buttonleisten

Buttonleisten können vielfältig eingesetzt werden und dienen dem Schnellzugriff auf diverse Funktionen. Wir unterscheiden 2 Typen von Buttonleisten, die Main Button Leiste und die applikationsbezogene Buttonleiste.



Die Main Button Leiste öffnet sich automatisch, sobald eine entsprechende Definitionstabelle vorliegt und Sie den Mauszeiger in die untere linke Bildschirmecke bewegen. Ebenso kann die Buttonleiste mittels der Tastaturkombination [APFEL] + [M] aufgerufen werden. Die Auswahl der einzelnen Funktionen kann dann mittels Mausklick auf den entsprechenden Button oder durch Betätigen der Zahlen Tasten „1" – „0" ausgelöst werden. Es sind maximal 10 Buttons je Leiste möglich. Die Skalierung der Buttongröße erfolgt automatisch.

Applikationsbezogene Button Leisten sind in ihrem Verhalten identisch der Main Button Leiste. Wenn Buttons definiert sind und sich die jeweilige Applikation im Vordergrund befindet, weckt die Positionierung des Mauszeigers in die untere rechte Ecke (oder die Tastenkombination [ALT]+[APFEL]+[M] die entsprechende Leiste auf.5.4.1 Aufbau der GUI Definitionstabellen (Standard Button Leisten)

Die GUI Definitionstabellen unterliegen einem einfachen Aufbau. Es sind reine csv Dateien mit einem „Semikolon" als Trennzeichen.

Die GUI-Definitionen müssen sich im Ordner:

[home]/Dokumente/macmyhome/gui

Der Aufbau der Dateien gestaltet sich wie folgt:

{ButtonName1};{Funktion};{ext.Option};{ext.Option};{ButtonImage};

{ext.Option} = in Standard Buttonleisten nicht verfügbar.

Button-Image ist optional. Zu kleine Bilder werden auf Button-Größe gestreckt (bei IMAGE=SCALE). Wenn ein Image angezeigt wird, wird der Text ebenfalls angezeigt. Die Bilder gehören ebenfalls in den gui-Ordner und müssen ohne Pfad angegeben werden. Es sind maximal 10 Buttons (= 10 Zeilen) möglich.

Ein entsprechender Eintrag würde sich wie folgt darstellen:

VISU EIN;<SCRIPT><RUN=Button_GUI_ON.myh></SCRIPT>;;;logo1.png

In diesem Fall würde ein Button mit dem Bildinhalt von "logo1.png" erzeugt, welcher bei Auslösung ein Script mit dem Namen „Button_GUI_ON.myh" ausführen würde.

VOL+;<SYS><MUTEOFF><VOLUP=5></SYS>;

Dieser Eintrag erzeugt einen Button mit dem Namen "VOL+". Bei Auslösung wird das System Mute abgeschaltet und die Systemlautstärke um 5 Einheiten angehoben. Die Namensgebung der Dateien ist festgelegt und wie folgt definiert:

Buttons.csv	**- Main Buttonleiste**
TV-Buttons.csv	**- eyeTV Buttonleiste**
VLC-Buttons.csv	**- VLC Buttonleiste**
BROWSER-Buttons.csv	**- Safari Buttonleiste**
ITUNES-Buttons.csv	**- iTunes Buttonleiste**
DVD-Buttons.csv	**- DVD player Buttonleiste**

Nachfolgendes kleines Beispiel soll den möglichen Aufbau einer Definitionsdatei aufzeigen:

Buttons.csv

```
VISU EIN;<SCRIPT><RUN=Button_GUI_ON.myh></SCRIPT>;;;logo1.png
VISU AUS;<SCRIPT><RUN=Button_GUI_OFF.myh></SCRIPT>;
iTunes EIN;<SCRIPT><RUN=iTunesON_PL1.myh></SCRIPT>;
iTunes AUS;<SCRIPT><RUN=AllHideAndMute.myh></SCRIPT>;
TV ON;<SCRIPT><RUN=TVon.myh></SCRIPT>;
CAM EIN;<SCRIPT><RUN=Button_CAM_ON.myh></SCRIPT>;
CAM AUS;<SCRIPT><RUN=Button_CAM_OFF.myh></SCRIPT>;
VOL+;<SYS><MUTEOFF><VOLUP=5></SYS>;
MUTE;<SYS><MUTEON></SYS>;
VOL-;<SYS><MUTEOFF><VOLDN=5></SYS>;
```

TIP:

Es empfiehlt sich den Buttons stets Scriptfiles zu hinterlegen. „Main_Button_1.myh", „Main_Button_2.myh" usw. Sie können sich dann einmalig Buttonleisten definieren und die eigentliche Funktion per Script übergeben.

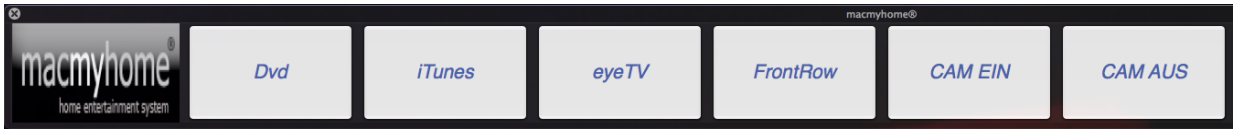
Bei der Verwendung von Bilddateien sollte beachtet werden, dass die Bildinhalte auch skaliert werden d.h. immer in Button Größe dargestellt werden.

5.4.2 Extended Button Leisten

Bei der Lizenzierung des GUI Moduls sind insgesamt 25 Buttonleisten möglich. Weiterhin können die Buttonleisten hinsichtlich Farbgebung, Textgröße etc. beliebig verändert werden. Buttonleisten können nun auch per Befehl aufgerufen und geschlossen werden. Neu ist auch, dass Buttonleisten nun auch mit 2facher Befehlsgebung (Toggler/Switch) konfiguriert werden können.

Dementsprechend hat sich der Aufbau der Definitionstabellen verändert.

Die Verwendung dieses Moduls ist abwärtskompatibel. Alte Definitionen können ohne weiteren Eingriff verwendet werden.



Die Namensgebung, sowie die reservierten Namen der Buttonleisten gilt, wie vor beschrieben. Der Dateiname Buttons.csv ist für die m..myhome Haupt-Buttonleiste reserviert. Speicherort ist nach wie vor der „[user]\dokumente\macmyhome\gui„- Ordner.

Buttonleisten sind zusätzlich auch links/rechts und oben darstellbar. Bitte beachten Sie jedoch, dass nur bei der Positionierung am unteren Bildschirmrand der volle Funktionsumfang der GUI zur Verfügung steht.

Aufbau der Definitionstabelle:

Die [Button].csv besteht aus 2 Sektionen:

[CONFIG];	*allgemeine Konfiguration der Buttonleisten-Parameter*
[BUTTONS];	*Definition von (max. 10) Buttons.*

[CONFIG]	*Sektion:*
TIMEOUT;{0...600}	*Timeout in Sekunden, nach dem die Buttonleiste nach Verlassen des Mauszeigers geschlossen wird, 0: Buttonleiste schließt nie automatisch, default: 5*
POSITION;{LEFT/RIGHT/TOP/BOTTOM}	*Position der Button Leiste (Default="BOTTOM")*
SIZE;{„pixel“/AUTO }	*Höhe der Button Leiste in Pixel bzw. Höhe= 1/10 des Bildschirms = AUTO (Default="AUTO")*

Hinweis:
HUD-Modus und Close-Button sind nicht möglich, wenn POSITION nicht BOTTOM ist!

CLOSEBUTTON;{YES/NO}	*definiert, ob oben links ein Closebutton dargestellt werden soll, default: NO*
HUD;{YES/NO}	*definiert, ob die Buttonleiste im HUD-Stil dargestellt werden soll, default: NO (nicht unter OSX Tiger)*
TRANS;{0.1...1.0}	*Transparenz der Buttonleiste, default: 0.8*

RED;{0.0...1.0}	*Rot-Anteil der Buttonleiste*
GREEN;{0.0...1.0}	*Grün-Anteil, wie vor*
BLUE;{0.0...1.0}	*Blau-Anteil , wie vor*
TEXTSIZE;{0...x}	*Zeichengröße der Button-Beschriftung, default: dynamisch, auflösungsabhängig*
TEXTRED;{0.0...1.0}	*wie oben, für die Schriftfarbe*
TEXTGREEN;{0.0...1.0}	*s.o.*
TEXTBLUE;{0.0...1.0}	*s.o.*
TEXTBACKGROUNDRED;{0.0...1.0}	*wie oben, für die Hintergrundfarbe der Buttons*
TEXTBACKGROUNDGREEN;{0.0...1.0}	*s.o.*
TEXTBACKGROUNDBLUE;{0.0...1.0}	*s.o.*
IMAGE;SCALE	*legt die Position des Button Bildes fest {SCALE (default) /CENTER/TOP/BOTTOM/LEFT/RIGHT}. SCALE skaliert das Bild auf Buttongröße*

Hinweise:

Wenn in der [CONFIG] -Sektion nichts definiert wird, verhält sich die Buttonleiste wie bisher. Die Farbkomponenten der 3 konfigurierbaren Farben müssen immer vollständig (alle 3 Komponenten mit gültigen Werten vorhanden) sein, sonst wird die Definition ignoriert. Wenn eine Button Hintergrundfarbe definiert ist, lässt sich kein Buttonrahmen darstellen und damit auch kein Zustand eines Toggle- Buttons.

[BUTTON] - Sektion:

Zeilenformat:

{ButtonName1};{Kommandosequenz1};{TOGGLE/TOGGLEON/INTERVAL=x[ms]/PUSHRELEASE/SWITCH/HOLD};{Kommandosequenz2 bei Button-Release};{ButtonImage1};{ButtonName2};{ButtonImage2}

{ButtonName2};{ButtonImage2} nur bei Verwendung des BUTTONTYP Switch

TIP:

Der „ButtonName“ oder auch nachfolgend Label genannt, kann auch Sonderzeichen wie zB ein CR oder LF enthalten. Hierüber kann der Labeltext mehrzeilig zur Anzeige gebracht werden. Sonderzeichen werden im Hexformat übergeben. Ein Eintrag „Mute|0x0DON/OFF“ würde eine zweizeilige Anzeige auslösen.

Beispiele:

MUTE;<SYS><MUTEON></SYS>

Erzeugt einen Standard-Button, der den Befehl „MUTEON" ausführt

MUTE;<SYS><MUTEON></SYS>;TOGGLE;<SYS><MUTEOFF></SYS>

Erzeugt einen Toggle-Button, der entsprechend seines Zustandes den entsprechenden Befehl ausführt

MUTE;<SYS><MUTEON></SYS>;TOGGLEON;<SYS><MUTEOFF></SYS>

Erzeugt einen initial betätigten Toggle-Button, der entsprechend seines Zustandes den entsprechenden Befehl ausführt. Der Button erscheint also nach Start von **m..myhome** im gedrücktem Zustand

MUTE;<SYS><MUTEON></SYS>;;;mute.jpg

erzeugt einen Standard-Button, der einen Befehl ausführt und ein Bild darstellt. Die Bilddatei muss sich ebenfalls im Ordner ..\gui befinden.

"VOL +";<SYS><VOLUP></SYS>;INTERVAL=200

Erzeugt einen Repeat-Button, der einen Befehl ausführt und wiederholt diesen bei gedrücktem Button alle 200ms

DIM+;<SCRIPT><RUN=DIMUPSTART.myh></SCRIPT>;PUSHRELEASE;<RUN=DIMUPSTOP.myh>

Erzeugt einen Flanken-Button, der einen Befehl/Script ausführt wenn der Button betätigt wird und führt ein weiteren Befehl/Script aus, wenn der Button wieder losgelassen wird.

MUTE ON;<SYS><MUTEON></SYS>;SWITCH;<SYS><MUTEOFF></SYS>;muteOn.jpg;MUTE OFF;muteOFF.jpg

Erzeugt einen SWITCH-Button, der beim 1. Druck die Aktion 1 ausführt und beim 2. Druck die Aktion 2 ausführt. Für die jeweiligen Zustände können verschiedene Befehle/Scripts sowie verschiedene Label/Bilder definiert werden. Wird keine Bilddatei angeben, werden die Texteinträge als Label übernommen. Die Bilddateien müssen sich ebenfalls im Ordner ..\gui befinden. Der Unterschied zum „TOGGLE-Button" liegt hier in der Verwendung von zustandsabhängigen Labeln/Bildern.

VOLUME;<SYS><VOLUP=5 ></SYS>;HOLD=2000;<SYS><VOLDN=5></SYS>;

Führt ein **VOLUP=5** aus, wenn der Button nach weniger als 2 Sekunden losgelassen wird, andernfalls wird nach 2 Sekunden sofort **VOLDN=5** ausgeführt auch wenn der Button weiterhin gedrückt wird.

Extended Buttonleisten Definitionen müssen wie folgt abgelegt werden:

{ButtonLeistenName}-Buttons.csv

{ButtonLeistenName} Beschreibt hierbei den internen Buttonleistennamen, der z.B. auch als Leistenüberschrift angezeigt wird. Verwenden Sie also den Dateinamen **„Filme-Buttons.csv"**, wenn Sie eine Leiste mit der Überschrift **„Filme"** erstellen möchten. Dieser Name wird ebenfalls benötigt, sofern Sie mit den nachfolgend beschriebenen Befehlen zur Steuerung der Buttonleisten arbeiten.

SYS-Kommandos zur Buttonsteuerung:

OPENBUTTONS={*ButtonLeistenName*} *öffnet eine Buttonleiste*

CLOSEBUTTONS={*ButtonLeistenName*} *schließt eine Buttonleiste*

SETBUTTONPUSHED={*ButtonLeistenName*},{1...10} *setzt den Zustand des gewählten TOGGLE- oder SWITCH-Buttons auf "gedrückt". Der Index zählt von links nach rechts.*

SETBUTTONRELEASED={*ButtonLeistenName*},{1...10} *wie oben, setzt den Zustand auf "nicht Gedrückt*

CLOSECURRENTBUTTONS *schließt die aktuell geöffnete Buttonleiste*

SETBUTTONLABEL={*ButtonLeistenName*},{1...10}{Label}
Ändert das Label eines Buttons flüchtig. Das so gesetzte Label verändert nicht den Eintrag in der Definitions-tabelle.

SETALTBUTTONLABEL={*ButtonLeistenName*},{1...10}{Label}
Ändert das 2. Label eines SWITCH Buttons flüchtig. Das so gesetzte Label verändert nicht den Eintrag in der Definitionstabelle.

DISABLEBUTTTON={*ButtonLeistenName*},{1...10} *schaltet einen Button ab und graut ihn aus. Der Button ist dann nicht mehr bedienbar*

ENABLEBUTTON={*ButtonLeistenName*},{1...10} *reaktiviert einen Button wieder, der zuvor abgeschaltet wurde*

.

Diese Befehle gelten auch bei Verwendung der Standard Buttonleisten. Bitte beachten Sie, dass die Namensgebung exakt wie beschrieben eingehalten wird!

Hinweise:

Die SETBUTTON...-Befehle funktionieren auch bei aktuell nicht sichtbaren Leisten. Wird eine neue Leiste bei bereits offener Buttonleiste geöffnet, wird die alte Leiste automatisch geschlossen.

Beispiele:

<SYS><OPENBUTTONS=DVD></SYS>

Öffnet die Definition DVD-Buttons.csv

<SYS><SETBUTTONLABEL=main,2,Mute/*0x0D*ON/OFF></SYS>

Tauscht das Label des Button "2" in der Main Buttonleiste.

Ausnahme/Sonderbedingung:

<SYS><OPENBUTTONS=macmyhome></SYS

Öffnet die Definition „Buttons.csv" (Main Buttonleiste). Die nicht eingehaltene Namensgebung gilt nur in diesem speziellen Fall.

Seit der Version 1.0.46 existiert ein externer Applikationsname "m..myhome", der z.B. für den Installer, das Config-Panel, den Titel der Main-Buttonleiste und das Copyright verwendet wird. Außerdem gibt es einen "internen" Applikationsnamen "macmyhome", der für Pfade und Dateinamen verwendet wird.

Um die Abwärtskompatibilität zu alten Scripten zu gewährleisten, reagiert die Main-Buttonleiste auf folgende Argumente von OPENBUTTONS und CLOSEBUTTONS:

"**macmyhome**"	(interner App-Name)
"**m..myhome**"	(externer App-Name = Titel)
"**main**"	(Rang der Buttonleiste)

Ohne Lizenz sind sämtliche Button- und Grafikfeatures abgeschaltet und es werden nur die Haupt-Buttonleiste sowie die applikationsabhängigen Buttonleisten eingelesen. Die Befehle OPENBUTTONS und CLOSEBUTTONS funktionieren jedoch weiterhin.

5.5. GIRA Homserver KO Gateway, KNXnet/IP Support

Im Nachfolgenden beschreiben wir die möglichen Schnittstellen für die EIB/KNX Anbindung. Diese Schnittstellen werden wir entsprechend der Markterfordernisse permanent anpassen und erweitern.

5.5.1 KNX IP BAOS 770

m..myhome unterstützt das BAOS (Bus Access and ObjektServer) Protokoll und ermöglicht somit die direkte Verarbeitung von bis zu 256 KNX Objekten je Mac System. Es ist dafür gedacht um einfache KNX Abläufe wie z.B. das Antriggern von Lichtszenen oder einfache Lichtschaltungen über die Buttonleisten einzubinden, oder eben um Aktionen auf dem Mac System (Script Aufruf oder Befehlsabläufe) anzutriggern. Um diese Schnittstelle nutzen zu können, ist das *Weinzierl KNX IP BAOS 770 Interface* zwingend notwendig. Die Konfiguration und die Definition der KNX Datenpunkte wird im BAOS Interface über die ETS (EIB Tool Software) vorgenommen. Die Vorgehensweise entnehmen Sie der Dokumentation des Herstellers.
Ist das Interface entsprechend eingerichtet, kann **m..myhome** mit dem Gerät Kommunizieren und Daten austauschen.

Hinweis:

Durch die Einbindung des KNXnet/IP Protokoll für Routing und Tunneling (ab Version 1.0.87) wird das BAOS Objekt Server Konzept nicht weiter verfolgt werden. Das Weinzierl KNX IP BAOS 770 Interface kann auch im KNXnet/IP Tunneling Modus betrieben werden.

Gehen Sie dabei wie folgt vor:

m..myhome erwartet in **../Documents/macmyhome/misc** eine Datei **baos.csv** mit folgendem Inhalt:

Aufbau der Definitionstabelle:

Die **baos.csv** besteht aus 3 Sektionen:

[CONFIG];	*allgemeine Konfiguration der Interface Parameter*
[DATAPOINTS];	*Definition von Datenpunkten, die zus. auf den BC Port übertragen werden sollen (optional)*
[ACTIONS];	*Definition der Datenpunkte, die eine direkte Aktion (Script Aufruf) auslösen sollen*

Aufbau:

[CONFIG];	*Sektion:*
BAOSIP;{IP}/AUTO/OFF	*{IP}; IP des KNXBAOS (fest)* *AUTO, IP des BAOS wird autom. im aktuellen Subnet gesucht* *OFF, BAOS-Funktion abgeschaltet*

optional zusätzlich (für den Empfang oder Script Aufruf notwendig):

[DATAPOINTS];	*Sektion:*
9;Klingel;HEX/DEC/ASCII	*{Nummer des BAOS-Datenpunkts};{frei wählbarer MMH-Name des Datenpunktes};{Darstellungsmodus}*

Beispiel:

9;Klingel;HEX	*generiert die BC - Ausgabe <BAOS>Klingel='VALUE'</BAOS>*

[ACTIONS];	*Sektion:*
9;0A;script.myh;HEX/DEC/ASCII	*Nummer;*{Datenpunkt};{Wert}/#; *{Scriptname};DEC/HEX/ASCII*

Beispiele:

9;0A;script.myh;HEX	*Führt das Script script.myh aus, wenn Datenpunkt 9 den Wert 10 hat*
9;10;script.myh;DEC	*exakt wie vor beschrieben*
9;ON;script.myh;ASCII	*Führt das Script script.myh aus, wenn Datenpunkt 9 der ASCII-Sequenz "ON" entspricht*
9;#;script.myh;DEC	*Führt das Script script.myh aus, wenn Datenpunkt 9 den Wert ändert und übergibt den Wert dezimal als [ARG] an das Script.*
9;#;script.myh;HEX	*Führt das Script script.myh aus, wenn Datenpunkt 9 den Wert ändert und übergibt den Wert als [ARG] hexadezimal an das Script*
9;#;script.myh;ASCII	*Führt das Script script.myh aus, wenn Datenpunkt 9 den Wert ändert und übergibt die ASCII-Sequenz als [ARG] an das Script*

Sobald Datenpunkte definiert sind, baut der Daemon eine feste, dauerhafte Verbindung zum BAOS auf und gibt empfangene Wertänderungen der in der .csv aufgelisteten Datenpunkte sofort über den Broadcast-Port aus. Über den Eventserver kann dann über eine entsprechende Definition eine entsprechende Aktion ausgelöst werden.

Generell werden die Datenpunkte immer über ihre Objeknummer angesprochen, also nicht über die entsprechende EIB/KNX Gruppenadresse, sondern über das Objekt mit dem die Adresse assoziiert wurde. Anstelle des Aufrufs einer Script Datei können auch direkte Befehlsketten (Kommandosequenzen) verwendet werden.
Beispw.: 9;1;<SYS><MUTON></SYS>;DEC

Die <BAOS> Klasse verfügt über folgenden Kommandosatz:

GETINFO	*liefert allgemeine Informationen über den BAOS*
GETDESCRIPTION={Datenpunktnummer}	*liefert die Einstellungen des gewählten Datenpunktes*
GETDESCRIPTIONSTRING={Datenpunktnummer}	*liefert die Beschreibung des gewählten Datenpunktes*
GETVALUE={Datenpunktnummer}	*liefert den aktuellen Wert des Daten -punktes inkl. Zustandsflags im HEX-Format*
SETVALUE={DatenpunktNr.},{HEX/DEC/ASCII}	*setzt den Wert des gewählten Datenpunkt in HEX, Dezimal oder als ASCII*
SENDVALUE={DatenpunktNr.},{ HEX/DEC/ASCII}	*sendet den Wert auf den KNX-Bus*

SETANDSENDVALUE={DatenpunktNr.},{ HEX/DEC/ASCII}
Kombination von SETVALUE und SENDVALUE

READVALUE={DatenpunktNr.},{ HEX/DEC/ASCII} *liest den Wert vom KNX-Bus*

CLEARTRANSMISSIONSTATE={DatenpunktNr.},{HEX/DEC/ASCII}
löscht den Verbindungsstatus

GETPARAMETERBYTE={ParameterNr.) *liefert das Byte des gewählten Parameters*

Beispiele:

<BAOS><SETANDSENDVALUE=4,1,DEC></BAOS>

Setzt das BAOS Objekt 4 auf den Wert 1

<BAOS><GETINFO></BAOS>

Frag die Parameter des BAOS Interface ab. Die Ausgabe erfolgt auf dem BC Port.

Wenn als Typ ASCII angegeben wird, wird der String auf 14 Bytes erweitert bzw. abgeschnitten

Mit der Version 1.0.59 wurden die „SET“ – Befehle um ein 3. Argument „HEX/DEC/ASCII“ erweitert. Wird kein Argument angegeben, wird von einem Dezimal Wert ausgegangen. Bei dem Typ „HEX“ ist „0x“ als Prefix nicht mehr notwendig. Die neue Typinterpretation ist jedoch abwärtskompatibel.

5.5.2 GIRA Homeserver KO-Gateway Support

Mit der m..myhome Version 1.0.59 unterstützen wir nun auch den bidirektionalen Zugriff auf das GIRA Homserver KO-Gateway. Diese ermöglicht die einfache Datenübergabe, da der Austausch von Informationen bis hin zum Zugriff auf den EIB/KNX direkt über das Gateway erfolgen kann. Die Kommunikation zwischen m..myhome und dem GIRA Homeserver muss dann nicht mehr zwingend über UDP/TCP Kommunikation erfolgen.

Bitte beachten Sie, dass die Datenpunkte, die Sie auswerten/ansprechen möchten im GIRA Homeserver für die Kommunikation über das KO-Gateway freigegeben wurden.

HS-Support:

Im misc Ordner eine Datei hs.csv anlegen.

Aufbau der Definitionstabelle:

Die **hs.csv** besteht aus 2 Sektionen:

[CONFIG]; *allgemeine Konfiguration der Interface Parameter*

[DATAPOINTS]; *Definition von Datenpunkten, die vom GIRA Homeserver empfangen und ausgewertet werden sollen.*

Aufbau:

[CONFIG];	*Sektion:*
HSIP;{IP Adresse des HS}	*IP Adresse des GIRA Homservers*
HSPORT;{KO-Portnummer auf dem HS}	*Port des GIRA KO - Gateway*
HSKEY;{ein evtl. im HS festgelegtes Passwort}	*Passwort/Key des KO-Gateway*
ADDRTYPE;{Adresstyp}	*Legt fest, wie die Adressausgabe im Broadcast erfolgen soll. Mögliche Werte „2STEP" oder 3STEP" leer oder nicht definiert = Dezimalausgabe.*

optional / zusätzlich (für den Empfang oder Script Aufruf notwendig):

[DATAPOINTS]; *Sektion:*

{KO im GIRA Homeserver};{Datenpunkt-Name};{Bedingung};{mmh-Sequenz }

Erläuterungen:

{KO im GIRA Homeserver}	*Adresse des Kommunikationsobjekts*
{Datenpunkt-Name}	*darf leer sein und ist frei definierbar*

{Bedingung}	kann (entsprechend BAOS) folgende Zustände haben:
{Match-String}	*der vom HS übermittelte Wert muß mit {Match-String} übereinstimmen*
{#}	*alle Werte starten das Script bzw. die Sequenz. wenn {Bedingung} leer ist, wird '#' angenommen.*
{mmh-Sequenz}	*Auszuführende Kommandosequenz oder Script, wenn Bedingung erfüllt. Kann als Platzhalter für den Wert \#' beinhalten, bei Scripten wird der empfangene Wert als Argument übergeben.*

Bitte beachten Sie, dass das KO-Gateway Werte, die keinen ASCII Text beschreiben, als Fließkomma Wert überträgt. So wird ein binäres „1“ oder „0“ als „1.0“ oder „0.0“ übergeben. Entsprechend ist das Feld {Bedingung} zu definieren.

Beispiele für die Sektion [DATAPOINTS]

8/1/4;iTunes Next;1.0;<ITUNES><NEXT><PLAY></ITUNES>

Bei Empfang der Adresse 8/1/4 mit dem Wert „1“ führt iTunes den internen Befehl „NEXT“ und PLAY aus. Auf dem Statusport wird die Meldung „iTunes Next=1.0“ generiert.

8/1/7;TV ON;1.0;<SCRIPT><RUN=StartTV.myh></SCRIPT>

Bei Empfang der Adresse 8/1/7 mit dem Wert „1“ wird das Script „StartTV.myh ausgeführt. Auf dem Statusport wird die Meldung „TV ON=1.0“ generiert.

5/1/0;Playlist;#;<SCRIPT><RUN=Play(Playlist).myh></SCRIPT>

oder, gleichwertig

5/1/0;Playlist;#;<ITUNES><PLAYPLAYLIST=\#></ITUNES>

Bei Empfang der Adresse 5/1/0 (EIS15 ASCII Definition) mit dem angenommenen Wert „Musik“ wird das Script „Play(Playlist).myh“ ausgeführt und der eingehende Text als Argument übergeben Auf dem Statusport wird die Meldung „Playlist=Musik“ generiert.

Neue Klasse HS mit folgenden Befehlen:

SETVALUE={Adresse},{Wert}	*setzt {Wert} an {Adresse} absolut*
SETVALUEREL={Adresse},{Wert}	*setzt {Wert} an {Adresse} relativ*
STEPUP={Adresse}	*erhöht den Wert an {Adresse}*
STEPDOWN={Adresse}	*senkt den Wert an {Adresse}*
LISTUP={Adresse}	*erhöht den Wert an {Adresse}*
LISTDOWN={Adresse}	*senkt den Wert an {Adresse}*

Statt {Adresse} kann bei allen Befehlen auch der in der .csv eingetragene Datenpunkt-Name verwendet werden. Wenn der HS innerhalb von 3 Sekunden auf den Befehl antwortet, wird der empfangene Wert als Rückgabewert ausgegeben.

Außerdem werden alle vom HS empfangenen Werte über den Broadcast-Port gesendet:
<HS><{Adresse bzw. Name, wenn definiert}={Wert}><.....></HS>

5.5.3 KNXnet / IP (Routing und Tunneling)

Mit der **m..myhome** Version 1.0.86 unterstützen wir nun auch das KNXnet/IP. Diese Einbindung ist die zZ optimalste Schnittstelle zum KNX/EIB, da es keine Restriktionen in der Anzahl der Datenpunkte zugrunde legt und keine zusätzlichen Datenpunktdefinitionen, wie es zum Beispiel bei dem BAOS Objekt Server notwendig ist, vorgenommen bzw. parametriert werden müssen.

Es werden beide Protokolltypen, also KNXnet/IP Routing und KNXnet/IP Tunneling unterstützt. Ebenso ist es möglich, eine „TUNNELING_BRIDGE" einzurichten. Dieser Modus wird im nachfolgenden noch näher erläutert.

Die Datenpunktdefinitionen werden in einer gesonderten Datei (knx.esf) im Ordner misc abgelegt. Diese Datei unterliegt dem OPC Export Format der ETS und kann direkt aus der ETS heraus exportiert werden. Es werden alle Datenpunkttypen unterstützt. Ebenso werden die dort definierten Namen für die Broadcast Ausgabe und im Logging Monitor verwendet. Die Typenkonvertierung und Skalierung erfolgt somit automatisch und muss nicht weiter berücksichtigt werden. Es kann auch ein individueller Name für die esf Datei vergeben werden. Dies ist in der CONFIG Sektion der knx.csv einstellbar.

Bei unbekannten Datentypen in der esf Datei erfolgt ein Log-Eintrag und die entsprechende Definition wird ignoriert. Datenpunkte, die nicht in der esf Datei eingetragen sind, werden ignoriert.

Besonderheit:
Da die OPC Exportschnittstelle Fehler behaftet ist, (2 Byte float Werte, wie zB Temperaturen werden als „uncertain", also undefiniert, übergeben) haben wir eine Anpassung vorgenommen. So werden alle 2-Byte uncertain Datenpunkte automatisch als „EIS 5 'Value' (2 Byte)" interpretiert.

5.5.3.1 KNXnet / IP Routing Support

Allgemeines:

Beim Start wird automatisch ein KNX/IP Routing-fähiger IP-Router gesucht und gegebenenfalls im Log angezeigt. **m..myhome** liest die Mutlicast-Adresse des IP-Routers aus und tritt der entsprechenden Gruppe bei.

Vorteil der Routing Variante: Es können beliebig viele Mac Systeme gleichzeitig auf das KNXnet / IP Device gleichberechtigt zugreifen.

Die Konfigurationsdatei heisst knx.csv und wird im misc-Ordner abgelegt. In der knx.csv angelegte Adressen werden konvertiert, ggfs. Actions ausgeführt und über den Broadcast gesendet.

Im misc Ordner eine Datei knx.csv anlegen.

Aufbau der Definitionstabelle:

Die **knx.csv** besteht aus 2 Sektionen:

[CONFIG]; *allgemeine Konfiguration der Interface Parameter*

[DATAPOINTS]; *Definition von Datenpunkten, die über KNXnet/IP empfangen und ausgewertet werden sollen.*

Aufbau:

[CONFIG]	*Sektion*
ACTIVE;YES	*Schaltet den KNXnet/IP Support „YES" = ein, „NO" = aus*
DEBUG;YES	*Schalter für „YES" =erweiterte Logausgaben (Busmonitor)„NO"* **=** *standard Logausgaben*
ESF;{ESF File}	*Name des esf File in dem Ordner misc (knx.esf = default)*
KNXADDR;{Phys.Adresse}	Lokale *Physikalische Absenderadresse, die mmh beim Versenden verwenden soll.*
ADDRTYPE;{Adresstyp}	**„***2STEP"=2 stufige Darstellung, „3STEP"=3 stufige Darstellung (Default)*

Spezifische Einstellungen für die Betriebsart „ROUTING"

CONNECTIONTYPE;{Protokolltyp}	**„***TUNNELING" oder „ROUTING" (default: ROUTING) oder* "TUNNELING_BRIDGE"

Bei CONNECTIONTYPE;ROUTING wird folgender zus. Schalter benötigt:

MULTICAST_IP; {Multicast ADR}	*{Multicast IP des Gerätes Standard: 224.0.23.12} oder „AUTO" für automatisches Suchen (default: AUTO)*

[DATAPOINTS];

{KNX/EIB Gruppenadresse};{Bedingung};{mmh-Sequenz}

Erläuterungen:

{KNX/EIB Gruppenadresse}		*Gruppenandresse im entsprechend eingestelltem Format*
{Bedingung}		*kann (entsprechend BAOS) folgende Zustände haben:*
	{Match-String}	*der von KNX/IP übermittelte Wert muß mit {Match-String} übereinstimmen.*
	{#}	*alle Werte starten das Script bzw. die Sequenz. wenn {Bedingung} leer ist, wird '#' angenommen.*
{mmh-Sequenz}		*Auszuführende Kommandosequenz oder Scriptname, wenn Bedingung erfüllt. Kann als Platzhalter für den Wert \#' beinhalten, bei Scripten wird der empfangene Wert als Argument übergeben*

Beispiele für die Sektion [DATAPOINTS]

8/1/4;1;<ITUNES><NEXT><PLAY></ITUNES>

Bei Empfang der Adresse 8/1/4 mit dem Wert „1" führt iTunes den internen Befehl „NEXT" und PLAY aus. Im Broadcast wird die Meldung „8/1/4 – [Name aus ESF]=1" generiert.

Pro Adresse lassen sich mehrere Actions definieren, wenn unterschiedliche Match-Bedingungen angegeben werden. Bei identischen Match-Bedingungen pro Adresse wird nur die erste gefundene Action ausgeführt.

Beispiel:

15/7/10;1;<SYS><SAY=on></SYS>
15/7/10;0;<SYS><SAY=off></SYS>

Es existiert eine neue Befehlsklasse KNX mit folgenden Befehlen:

SETVALUE={KNX-Adresse},{Wert} - *sendet ein Telegramm entsprechend der Definition aus der .esf-Datei auf den Bus. Support für Uncertain-Typen (1,2,3,4,8 Bit und 1,2,3,4 Byte) - mehrere Bytes werden beim SETVALUE-Befehl durch Kommata getrennt*

GETVALUE={KNX-Adresse} - *fragt den Wertinhalt der KNX-Adresse ab Das Rückgabeformat lautet: GETVALUE={GA}-{Name}={Wert}, also sehr ähnlich zu den Broadcast-Ausgaben. Es wird maximal eine Sekunde auf eine Antwort gewartet.*

5.5.3.2 KNXnet/IP Tunneling Support

Allgemeines:

Für den Tunneling Support gelten gleiche Vorgehensweisen, wie zum Routing Protokoll beschrieben. Es müssen lediglich nachfolgende Schalter in der CONFIG Sektion der knx.csv angepasst bzw. verändert werden.

Die Betriebsart TUNNELING lässt lediglich nur eine aktive Verbindung zu dem KNXnet/IP Device zu. Dies kann sich grad bei größeren Installationen nachteilig auswirken. Eine Möglickeit diesen „Nachteil" zu umgehen bieten wir Ihnen mit der Betriebsart **„TUNNELING_BRIDGE"**, die im nachfolgenden Absatz beschrieben wird.

Änderungen für den Tunneling-Support:

Spezifische Einstellungen für die Betriebsart „TUNNELING"

CONNECTIONTYPE;TUNNELING

Bei CONNECTIONTYPE;TUNNELING wird folgender zus. Schalter benötigt:

DEVICE_IP; {Gateway IP} *legt die IP des KNX-Tunneling-Gerätes fest, mögliche Werte: {IP des Gerätes} oder „AUTO" für automatisches Suchen (default: AUTO)*

5.5.3.3 KNXnet/IP TUNNELING_BRIDGE Support

Allgemeines:

Der KNXnet/IP Tunneling Bridge Support dient dazu mehrere Mac Systeme über lediglich eine bestehende Tunneling Verbindung den Zugriff in die KNX Welt zu ermöglichen. In der Betriebsart „TUNNELING_BRIDGE" arbeitet der Mac quasi als virtueller KNX/IP Router, der die KNX Verbindung über eine Tunneling Verbindung Sicherstellt. Diese Funktionalität kann zB auch zur Anlagenkopplung verwendet werden.

Vorgehensweise:

Ausgehend von min. 2 Mac Systemen mit aktiverter KNXnet/IP Einstellung und einem KNXnet/IP Tunneling Device müssen folgende Einstellungen vorgenommen werden:

Ein Mac im System muss als „Tunneling Master" arbeiten. Dieser stellt die Tunneling Verbindung zu dem KNXnet/IP Device sicher. Gleichzeitig stellt dieser Mac eine Multicast Adresse für den Routing Support bereit. Über diese Multicast Adresse können nun weitere Mac's mit dem „Tunneling Master" kommunizieren. Die weiteren Mac's arbeiten hierfür in der Betriebsart „Routing".

Bitte stellen Sie sicher, dass die Multicastadresse bzw. der dadurch definierte Multicast Bereich nur den Mac Systemen zur Verfügung steht. Sollte ein weiteres KNXnet/IP Routing fähiges Device in der Gesamtstruktur erreichbar sein, kann dies zu Störungen bei der Kommunikation führen. Es ist auch darauf zu achen, dass in den Einstellungen keine „AUTO" Funktion aktiviert ist.

Konfiguration „Tunneling Server" (1.Mac)

knx.csv - nachfolgende Einstellungen beispielhaft.

[CONFIG];
ACTIVE;YES
DEBUG;YES
ESF;knx.esf
PHYSADDR;0.0.254

//** Einstellung für TUNNELING_BRIDGE ****

CONNECTIONTYPE;TUNNELING_BRIDGE - *Schalter für Betriebsart*

DEVICE_IP;AUTO - „AUTO" sucht das KNXnet/IP Device, besser ist jedoch die Einstellung einer festen IP Adresse

MULTICAST_IP;224.0.23.15 - Multicast Adressbereich worüber der Routing Service bereit gestellt wird.

Diese Einstellungen definieren das KNXnet/IP Interface des Mac Systems als Multicast Gateway mit Tunneling Zugriff.

Konfiguration „Routing" (2. bis n. Mac System)

knx.csv - nachfolgende Einstellungen beispielhaft.

[CONFIG];
ACTIVE;YES
DEBUG;YES
ESF;knx.esf
PHYSADDR;0.0.253

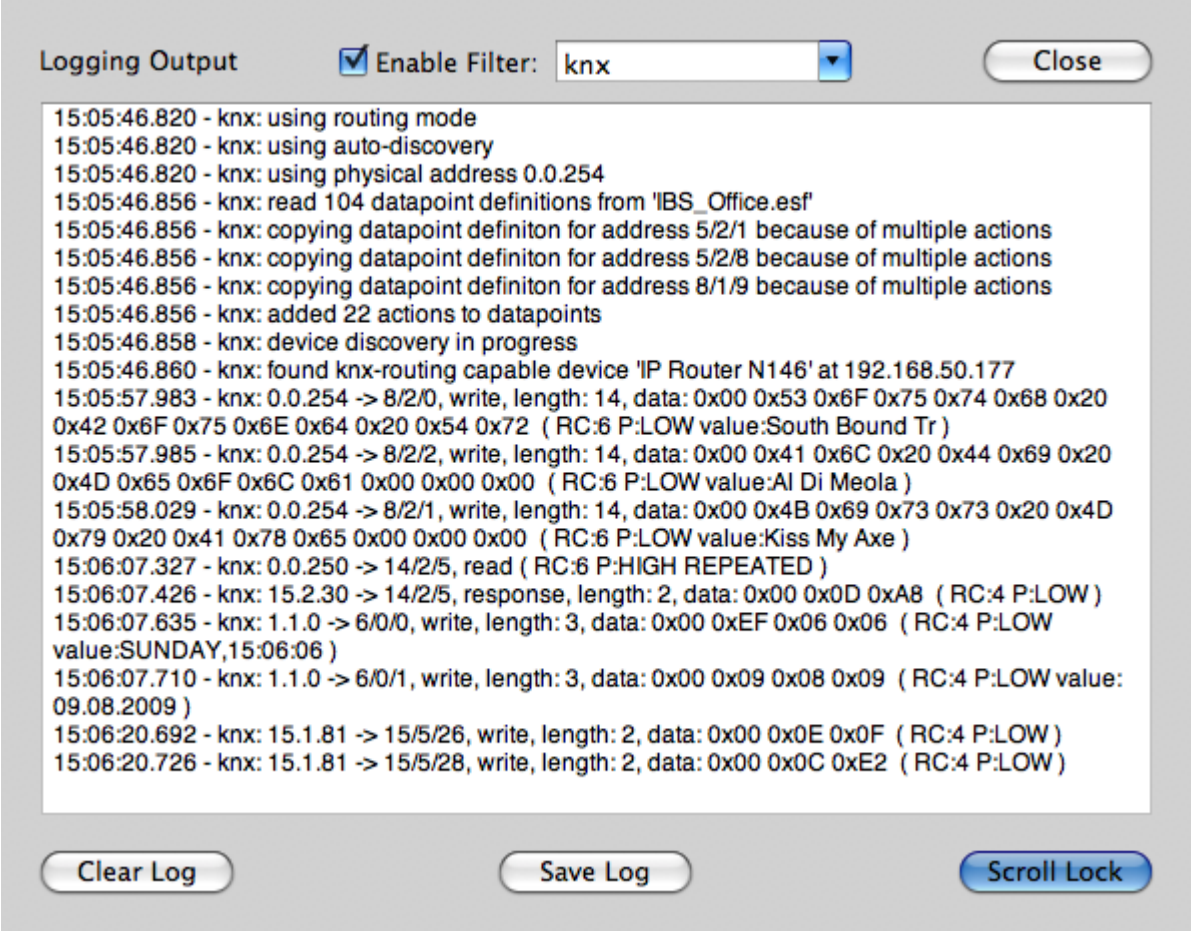
//** Einstellung für ROUTING ******

CONNECTIONTYPE;ROUTING - *Schalter für Betriebsart*

MULTICAST_IP;224.0.23.15 - Multicast Adressbereich worüber der Routing Service bereit gestellt wird.

Alle Geräte im Netz, die das Routing-Protokoll unterstützen, können nun über diese „**MULTICAST_IP**“ 224.0.23.15 eine Gateway Verbindung zu dem „Tunneling Master“ aufbauen und gleichzeitig auf den KNX-Bus zugreifen.

Bitte beobachten Sie nach einem Neustart auch die Logausgaben der jeweiligen Mac Systeme. Das Log gibt Auskunft über den jeweiligen Verbindungsstatus


Logging Output
Enable Filter:
knx
Close
15:05:46.820 - knx: using routing mode
15:05:46.820 - knx: using auto-discovery
15:05:46.820 - knx: using physical address 0.0.254
15:05:46.856 - knx: read 104 datapoint definitions from 'IBS_Office.esf'
15:05:46.856 - knx: copying datapoint definiton for address 5/2/1 because of multiple actions
15:05:46.856 - knx: copying datapoint definiton for address 5/2/8 because of multiple actions
15:05:46.856 - knx: copying datapoint definiton for address 8/1/9 because of multiple actions
15:05:46.856 - knx: added 22 actions to datapoints
15:05:46.858 - knx: device discovery in progress
15:05:46.860 - knx: found knx-routing capable device 'IP Router N146' at 192.168.50.177
15:05:57.983 - knx: 0.0.254 -> 8/2/0, write, length: 14, data: 0x00 0x53 0x6F 0x75 0x74 0x68 0x20 0x42 0x6F 0x75 0x6E 0x64 0x20 0x54 0x72 (RC:6 P:LOW value:South Bound Tr)
15:05:57.985 - knx: 0.0.254 -> 8/2/2, write, length: 14, data: 0x00 0x41 0x6C 0x20 0x44 0x69 0x20 0x4D 0x65 0x6F 0x6C 0x61 0x00 0x00 0x00 (RC:6 P:LOW value:Al Di Meola)
15:05:58.029 - knx: 0.0.254 -> 8/2/1, write, length: 14, data: 0x00 0x4B 0x69 0x73 0x73 0x20 0x4D 0x79 0x20 0x41 0x78 0x65 0x00 0x00 0x00 (RC:6 P:LOW value:Kiss My Axe)
15:06:07.327 - knx: 0.0.250 -> 14/2/5, read (RC:6 P:HIGH REPEATED)
15:06:07.426 - knx: 15.2.30 -> 14/2/5, response, length: 2, data: 0x00 0x0D 0xA8 (RC:4 P:LOW)
15:06:07.635 - knx: 1.1.0 -> 6/0/0, write, length: 3, data: 0x00 0xEF 0x06 0x06 (RC:4 P:LOW value:SUNDAY,15:06:06)
15:06:07.710 - knx: 1.1.0 -> 6/0/1, write, length: 3, data: 0x00 0x09 0x08 0x09 (RC:4 P:LOW value: 09.08.2009)
15:06:20.692 - knx: 15.1.81 -> 15/5/26, write, length: 2, data: 0x00 0x0E 0x0F (RC:4 P:LOW)
15:06:20.726 - knx: 15.1.81 -> 15/5/28, write, length: 2, data: 0x00 0x0C 0xE2 (RC:4 P:LOW)
Clear Log
Save Log
Scroll Lock

5.6. Airport Support (Airfoil)

Mit der **m..myhome** Version 1.0.46 wurde die Steuerung externer Lautsprecher über den Apple Airport Express eingebunden. Die Nutzung dieser Funktion setzt den Einsatz der Arfoil Software voraus. Die Software kann unter folgender URL bezogen werden: *http://rogueamoeba.com/airfoil/mac/*. Die Installationshinweise finden Sie ebenfalls auf der Homepage des Herstellers. Bitte beachten Sie, dass die Anwendungen, deren Ton übertragen werden soll, entsprechend in den Airfoil Einstellungen angemeldet werden müssen (erscheinen im Pull-Down_Menü von Airfoil).

Mit der Airport Integration wurde die **<SYS>** Klasse des **m..myhome** Protokoll entsprechend erweitert und verfügt nun über folgendem zusätzlichen Kommandosatz:

GETSPEAKERLIST	*listet alle verfügbaren Airports mit Namen auf ("Computer" ist der lokale Ton des Macs)*
GETSPEAKERSOURCE	*gibt die aktuelle Applikation bzw. den aktiven Eingang aus, deren Ton gerade gestreamt* wird
SETSPEAKERSOURCE={Appname}	*wählt die Applikation aus, deren Ton gestreamt werden soll*
SETSPEAKERSOURCEDEVICE={Eingang}	wählt den Audio-Eingang aus, deren Ton gestreamt werden soll
GETAUDIODEVICELIST	*listet alle verfügbaren Audio-Eingänge auf*
GETSPEAKERVOL={Airport-Name}	*gibt die aktuelle Lautstärke des gewählten Airports aus*
SETSPEAKERVOL={Airport-Name,}{0...100}	*setzt die Lautstärke des gewählten Airports absolut, wird der Name weggelassen, wird die Lautstärke für alle Airports (inkl. "Computer") gesetzt*
SPEAKERVOLUP={Airport-Name,}{0...100}	*erhöht die Lautstärke um den angegebenen Wert, der Name ist auch hier optional*
SPEAKERVOLDN={Airport-Name,}{0...100}	*senkt die Lautstärke, sonst wie SPEAKERVOLUP*
ENABLESPEAKER={Airport-Name}	*schaltet den gewählten Airport ein*
DISABLESPEAKER={Airport-Name}	*schaltet den gewählten Airport aus*
ENABLEALLSPEAKERS	*schaltet alle Airports ein*
DISABLEALLSPEAKERS	schaltet alle Airports aus
GETSPEAKERSTATE={Airport-Name}	liefert den Status des gewählten Airports: ENABLED / DISABLED / NOT_AVAILABLE

Beispiele:

Hinweise:

Wir weisen ausdrücklich darauf hin, dass mit dieser Lösung keine Multiroom Anwendungen zu realisieren sind. Die Airport Steuerung dient lediglich der einfachen Einbindung von max. 3-4 zusätzlichen Audiozonen. Zu beachten ist auch, dass der empfangene Stream etwas zeitversetzt gesendet wird. Es ist zB nicht möglich, den Audiostream zu synchronisieren. Die Audio Wiedergabe von zb TV Sendungen oder anderen Anwendungen mit Audio/Video Übertragungen ist durch diesen Zeitversatz nicht Lippensynchron und daher nicht zu empfehlen.

5.7. eyeTV - Streaming Support

Sofern das System über einen TV Tuner verfügt, besteht die Möglichkeit den unter eyeTV abgespielten TV Kanal in das Netzwerk zu streamen. Der Stream kann z.B. über die kostenlose Software VLC empfangen werden. Das notwendige eyeTV Plugin wir automatisch mit der **m..myhome** Installationsroutine installiert.

Die Steuerung des Streaming Servers erfolgt in der der Klasse STREAMER mit folgenden Befehlen:

ON	*startet die Streaming-Engine und ggfs EyeTV (minimiert)*
OFF	*beendet die Streaming-Engine, EyeTV läuft weiter*
CH={Kanalnummer} oder {Kanalname}	*wählt die Streaming-Quelle, dieser Befehle funktioniert analog zum SELECT-Befehl in der Windows-Klasse. Alle nachfolgenden, kanalbezogenen Befehle beziehen sich auf den zuletzt mit CH ausgewählen Kanal*
GETCHLIST	*liefert eine Liste mit allen zur Zeit verfügbaren Kanälen und der entsprechenden Kanalnummer.*
ADDTARGET={IP:Port}	*weist der zuvor mit CH gewählten Quelle ein Ziel zu und beginnt sofort mit dem UDP Streaming zur angegebenen Adresse. Es wird eine eindeutige Streaming-ID zurückgegeben. Auf dem Client dem VLC bitte als URL: udp://@:{Portnummer} übergeben.*
REMOVETARGET={IP:Port} oder {Streaming-ID} oder {IP}	*stoppt das Streaming des per IP:Port oder ID angegeben Streams. Wird nur die IP angegeben, werden alle an diese IP gerichteten Streams gestoppt.*
REMOVEALLTARGETS	*stoppt alle Streams*
GETTARGETSFORIP={IP}	*listet alle an die IP gerichteten Streams auf*
GETTARGETSFORCH={Kanalnummer} oder {Kanalname}	*listet alle IP-Adressen auf, an die der Kanal gestreamt wird*
GETALLTARGETS	*listet alles auf, was gestreamt wird*

Grundsätzlich:

Die Kanalnummern der jeweiligen Sender sind nicht festgelegt, sondern können sich beim Neustart von EyeTV oder **m..myhome** ändern. Deshalb ist es sinnvoller, den Sender anhand des Namens auszuwählen.
Es stehen immer alle Sender eines Transponders zur Verfügung. Mit 4 Tunern sind so also 16 Programme gleichzeitig möglich, wenn man die Sender in EyeTV geschickt wählt. Ein USB-Stick im Dual-Mode liefert also 8 Programme gleichzeitig. Sollte es bei mehr als 2 TV-Bildern in EyeTV zu Streaming Aussetzern kommen, liegt das an der hohen Prozessorlast, die EyeTV erzeugt. Wir empfehlen dann Einfach die nicht lokal notwendigen TV-BIlder ins Dock minimieren.

Hinweis:

Streaming an Broadcast-Adressen ist möglich. Die Zahl der Streams ist nur durch die MAC-Hardware bzw. das Netzwerk beschränkt.Zusätzlich ist TCP-Streaming ohne Steuerungsbefehle möglich: die Portnummer entspricht dabei der Kanalnummer+61000. Wenn z.b. RTL auf Kanal 5 läuft, kann der VLC sich dieses Programm mit der URL tcp://{Mac-IP}:61005 "abholen". Die Zahl der TCP-Streams ist ebenfalls nicht begrenzt. Sollte es beim TCP-Streaming zu kurzen (Ton-)Aussetzern kommen, bitte in den Einstellungen des VLC das Caching nach oben setzen, z.B. auf 500ms.

Beispiele:

Aktivieren der Streaming-Engine (EyeTV wird hierbei ggfs. mitgestartet):

<STREAMER><ON></STREAMER>

Ausgeben aller z.Zt. verfügbaren DVB-Streams:

<STREAMER><GETCHLIST></STREAMER>

Antwort:
<STREAMER>GETCHLIST=CH1=Das Erste|CH2=Bayerisches FS Süd|CH3=hr-fernsehen|CH4=Bayerisches FS Nord|CH5=WDR Köln|CH6=BR-alpha*|CH7=SWR|OK</STREAMER>

UDP-Streaming des Programms „Das Erste" an IP 192.168.1.240, Port 45678:

<STREAMER><CH=Das Erste><ADDTARGET=192.168.1.240:45678></STREAMER>

Antwort:
<STREAMER>CH=1|Das Erste|OK|ADDTARGET=567739290|OK</STREAMER>

5.8. m..Remote Support (cf iViewer)

Der m..Remote Service wird in einer gesonderten Dokumentation ausführlich beschrieben.

-

6. Sonstiges

6.0. Hinweise zur Konfiguration Ihres Mac

Um Ihr Mac System optimal auf den Einsatz von **m..myhome** vorzubereiten, empfehlen wir folgende Vorgehensweise:

Erstellen Sie für die Verwendung von **m..myhome** einen eigenen Benutzer (Profil). Verwenden Sie bei der Namengebung einen möglich kurzen Profilnamen z.B. „mmh". Dies erleichtert die spätere Verwendung bzw. den Zugriff auf lokale Ressourcen.

Deaktivieren Sie sämtliche Funktionen, die undefinierte Dialoge wie z.B. den Hinweis auf Programmupdates auslösen. Derartige Dialoge verhindern bis zu deren Bestätigung die Abarbeitung empfangener Befehlssequenzen.

Projektdaten und Konfigurationstabellen befinden sich immer im Ordner [user] \Dokumente\macmyhome. Sichern Sie diesen Ordner mit allen sich darin enthaltenen Dateien. Sie können diesen Ordner auch für weitere Systeme verwenden. Bei jedem Neu- oder Restart/Reset des **m..myhome** Services wird der Inhalt des Ordners durchsucht und die darin enthaltenen Daten verarbeitet.

6.1. Verwendung von Applescript

Ab der Version 1.0.37 kann der **m..myhome** Daemon auch direkt über Applescript eingebunden werden. Dies ermöglicht die einfache Einbindung in Apple eigene Anwendungen. Software wie z.B. Remotebuddy kann entsprechend konfiguriert werden, so dass der **m..myhome** Service und deren Protokollstruktur ohne Einschränkungen genutzt werden kann.

Syntax:

```
Tell application „MyHGUI“
    Command „<SYS><MUTEON></SYS>“
end tell
```

6.2. Arbeiten mit Systemvariablen

Ab der Version 1.0.44 unterstützt **m..myhome** die Verwendung von Systemvariablen. Sie können unter Verwendung der ...**VAR** Befehle aus der **<SYS>** Klasse beliebige Werte speichern oder gespeicherte Werte setzen und auch verwenden. Dies ist bei größeren Installationen sinnvoll, da hierüber z.B. auch Script Aufrufe manipuliert, Lautstärken oder, Systemstatis definiert und gespeichert werden können. Speicherort für die Systemvariablen ist **../macmyhome/misc/sysvars.csv**. Die Datei wird automatisch angelegt, falls nicht vorhanden.
Die Werte der Variablen werden automatisch eingesetzt, wenn in der Kommandosequenz der String [{Variablenname}] gefunden wird.

Beispiel:

<SYS><SETVAR=STARTVOLUME,33></SYS>	*Speichert eine Variable Namens „Startvolume" mit dem Wert 33*
<SYS><SETVOL=[STARTVOLUME]></SYS>	*Setzt die Systemlautstärke auf den Wert der Variable „Startvolume"*
<SYS><GETALLVARS></SYS>	*Liefert die Ausgabe aller vorhandenen Variablen und deren Wertinhalt.*

Dynamische Systemvariablen

Ab der Version 1.0.66 werden sog. dynamische Systemvariablen automatisch generiert. Diese Variablen werden bei Systemstart definiert und dienen zB der automatischen Übergabe systemspezifischer Parameter wie Bildschirmauflösung, IP Adresse, URL des Projektpfades und vieles mehr.

[SYSTEM_IP]	*- aktuelle IP*
[SCREEN_X]	*- x-Auflösung des Bildschirms*
[SCREEN_Y]	*- y-Auflösung des Bildschirms*
[BUTTON_SIZE]	*- aktuelle Höhe der Buttonleiste*
[HOMEDIR]	*- aktuelles Verzeichnis*
[PROFILE]	*- aktuelle Profilname (Kurzname)*
[PROJECT]	*- aktueller Projektpfad*
[TIME]	*- aktuelle Systemzeit*
[DATE]	*- aktuelles Systemdatum*

Hinweis zu [TIME] und [DATE]:

Diese Variablen können auch mittels Systembefehl <SYS><GETTIME></SYS> und <SYS><GETDATE></SYS> abgefragt werden. Das Ausgabeformat ist so angepasst worden, dass der Wertinhalt direkt an ein KNX EIS3 Objekt übergeben werden kann.

Sämtliche Variablen können überschrieben werden, d.h. eine manuelle Definition geht vor. Der Befehl GETALLVARS liefert immer alle Variablen zurück, d.h. wenn z.b. SYSTEM_IP in der .csv definiert ist, kommt die Variable zweimal in der Antwortsequenz vor.

Durch die Einbindung der dynamischen Systemvariablen steht ein mächtiges Werkzeug zur Verfügung um Profile dynamisch aufzubauen. Nahezu sämtliche Einstellungen können mittels Systemvariablen übergeben werden. Auch CONFIG Einstellungen/Sektionen der CommandServer Definitionen, wie zB IP- oder Port- Adressen können mittels Systemvariablen übergeben werden. Somit ist es Möglich spezifische Einstellungen der Konfiguration allein aus der sysvars.csv zu steuern.

Achtung: Die Variablen werden nur beim Start von mmh gelesen, eine Änderung der Variablen zur Laufzeit hat keine Wirkung und wird erst nach einem Neustart übernommen! Ausgenommen: Variablen, die per SETVAR Befehl während der Laufzeit beschrieben werden.

Beispiel (webserver.csv):

DVD_PATH;[MY_PATH]

Und in der sysvars.csv den folgenden Eintrag:

MY_PATH;[HOMEDIR]/Movies

Durch den Einsatz der dynamischen Variable **[HOMEDIR]** wird die Variable „**MY_PATH**“ automatisch mit der URL des aktuell angemeldeten Users um den Eintrag „/Movies“ vervollständigt und übergeben. Dieser Eintrag kann selbstverständlich auch direkt in der Webserver.csv vorgenommen werden, jedoch würde es den Sinn einer „zentralen“ Konfigurationsdatei verfehlen.

Systemvariablen können sich automatisch wie Feldvariablen verhalten, wenn die Wertinhalte durch ein „|“ getrennt werden.

Beispiel:

<SYS><SETVAR=TEST,Wert1|Wert2|Wert3|Wert4></SYS>

Hierüber wird nun eine Variable „Test“ mit den Inhalten „Wert1“, „Wert2“, „Wert3“, „Wert4“ in die sysvars.csv geschrieben. Das Trennzeichen „|“ indiziert hierbei den Index des jeweiligen Werteintrages.

Die Abfrage mit einem Formatparameter:

<SYS><GETVAR{%I2,1}=Test></SYS>

Liefert nun den Wert des zweiten Eintrages der Variable „Test“ mit einer Länge von 1, also lediglich einen Eintrag, aus der vorherig angelegten Variablen. Das Ergebnis lautet entsprechend:

<SYS>GETVAR{%I2,1}=Wert2|OK</SYS>

Oder wie im folgenden eine Abfrage:

<SYS><GETVAR{%I1,2}=Test></SYS>

Mit der Antwort:

<SYS>GETVAR{%I1,2}=Wert1|Wert2|OK</SYS>

Ein gezieltes Schreiben auf einen Index ist z.Z. nicht möglich.

6.3 Mini- Webserver

In **m..myhome** ist ein kleiner Webserver eingebunden worden, dessen Aufgabe es ist, Informationsinhalte bereitzustellen. Diese Inhalte (Content) können Systemen von Drittanbietern zur Verfügung gestellt werden. So wird z.B immer das Cover des aktuell angespielten iTunes Song unter einer statischen Web Adresse bereitgestellt. Ebenso können Bildinformationen des aktuellen TV Senders (Sender Logo) oder das Cover der aktuell abgespielten DVD abgerufen werden.

Über den Webserver wird erfolgt zB auch die Bereitstellung des auf dem Mac System abgelegten mRemote Projekt zum Download auf das iPhone / iPod Touch (näher Hinweise hierzu finden Sie in der mRemote Dokumentation).

Der Webserver wird zukünftig weitere Möglichkeiten bieten Informationen bereit zu stellen und daher permanent weiterentwickelt.

Mini-Webserver:

Im misc-Ordner eine webserver.csv mit folgendem Inhalt anlegen:

[CONFIG];	
ACTIVE;YES	*- Aktiviert/Deaktiviert den Webserver „YES" (Default)= aktiv, „NO" = nicht aktiv*
DEBUG;NO	*- Erweiterte LOG Ausgabe „YES"= aktiv, „NO" (DEFAULT) = nicht aktiv*
PORT;{Portnummer}	*- Für {Portnummer} sollte man einen Port > 1024 (Standard) verwenden.*
DVD_PATH;{Pfad zu den DVD-Verzeichnissen}	*- Startpfad des DVD Verzeichnisses. Z.B „DVD_PATH;/Users/mmh/Movies/" für den lokalen Ordner „Filme"*

Anwendung:

iTunes Support:

Folgende URL's werden von dem Webserver bereit gestellt:

http://{IP}:{Portnummer}/itunes_current.jpg

Zeigt das Cover des laufenden iTunes Titels

http://{IP}:{Portnummer}/itunes_ID_{ID}.jpg

Zeigt das Cover des iTunes Titels nach Aufruf der ID

http://{IP}:{Portnummer}/itunes_album_{Name}.jpg

Zeigt das Cover des iTunes Album nach Aufruf des Album Namens.

DVD-Cover Support:

folgende Aufrufe sind möglich:

http://{MacIP}:{Portnummer}/dvd_current.jpg

Liest den aktuellen DVDTitel aus und sucht in {DVD_PATH}/{DVDTitel} nach einer Datei **"preview.jpg"** und liefert diese zurück. {DVDTitel} wird hierbei aus den Informationen der laufenden DVD automatisch generiert.

http://{MacIP}:{Portnummer}/dvd_{DVDTitel}.jpg

Sucht in {DVD_PATH}/{DVDTitel} nach einer Datei "preview.jpg" und liefert diese zurück. Beispiel: *http://192.168.50.55:1234/dvd_Casino%20Royale.jpg* html-Zeichen (%) werden automatisch konvertiert. „previewjpg" kann beliebig tief im zu durchsuchenden Ordner liegen

TV-Logo Support:

Im misc-Ordner einen Unterordner "web" anlegen, wenn nicht vorhanden. In diesem Ordner die Senderlogos als *.png oder *.jpg ablegen. Es ist auf folgende Namenskonvention zu achten:

tv_{Sendername}.jpg

{Sendername} sollte dabei exakt dem Namen entsprechen, der vom Befehl TV - GETCHNAME zurückgeliefert wird. Beispiel, Logoname im Verzeichnis „web":

tv_SAT.1.jpg oder **tv_SAT.1.png**

Die Logos lassen sich durch Angabe des Dateinamens vom Webserver herunterladen um z.B. eine Programmliste zu erstellen. Ausserdem gibt es auch hier die fest definierte **tv_current.jpg.** Und der Aufruf:

http://{MacIP}:{Portnummer}/tv_current.jpg oder **/tv_current.png**

…liest den aktuellen Sender aus und liefert das entsprechende Logo (wenn vorhanden).

Für alle Grafiken gibt es Fallbacks. Diese werden ausgegeben, wenn keine entsprechende Grafikdatei gefunden wird. Diese Dateien müssen ebenfalls im „web" – Ordner abgelegt und wie folgt benannt werden:

itunes_dummy.jpg, dvd_dummy.jpg, tv_dummy.jpg oder eben als .png Datei.

Groß- und Kleinschreibung spielt weder bei den Dateinamen noch bei den URL-Aufrufen eine Rolle.
Der Grafikpuffer wurde auf maximal 10MB definiert. Damit sollten sich alle Grafiken darstellen lassen. Es sollte jedoch darauf geachtet werden, je nach Anwendung, die Dateien klein zu halten und nur für die Darstellung notwendige Pixelgrößen zu verwenden.

- Feature Webserver: Bei Aufrufen mit '/' im Dateinamen wird der '/' durch ein blank ersetzt
- Feature Webserver: Kommt im TV-Programmnamen ein '/' vor, wird er durch ein blank ersetzt

6.4 Timer Support

m..myhome unterstützt Timer. Timer dienen zB der zyklischen Abfrage bestimmter Informationen. Timer können im laufenden Prozess gestartet oder gestoppt werden. Ebenso ist es möglich, Timer mit einem Autostart zu belegen.

Konfiguration:

Im **../Dokumente/macmyhome/misc/** Ordner eine Datei **timer.csv** anlegen:

[CONFIG]; *Sektion*

ACTIVE;YES *„YES"= aktiv, „NO" = nicht aktiv*

DEBUG;NO *„YES" erzeugt einen Logeintrag, wenn ein Timer feuert. Sonst werden nur Start bzw. Stopp eines Timers geloggt.*

[TIMERS]; *Sektion*
{Name};{Typ};{Wert};{Action};{Optionen}

{Name} *- Bezeichnung des Timers*

{Typ} *- Typ des Timers, "INTERVAL" oder „ONESHOT"*

{Wert} *- Konfiguration des Timerwertes, für INTERVAL-Timer: Intervall in Sekunden , für ONESHOT-Timer: Zeit in Sekunden, nach der der Timer einmalig feuert, (max. 86400 (= 1Tag) möglich)*

{Action} *- auszuführende mmh-Sequenz bzw. mmh-Scriptname*

{Optionen} *- weitere Optionen zur Timerkonfiguration, z.Zt. möglich:*
„AUTOSTART"*- startet den Timer beim Start von* **mmh** automatisch

Beispiel:

[TIMERS];

IVT;INTERVAL;1;<ITUNES><GETPTIME><GETPTIME{%02m:%02s}><GETCTIME><GETCTIME{%02m:%02s}><GETREPEAT><GETSHUFFLE></ITUNES>;AUTOSTART

Startet den Timer **IVT** automatisch und führt die Itunes-Befehle im Ein-Sekunden-Takt aus.

CLEARSCREEN;ONESHOT;5;<WINDOW><SELECT=3><RELEASE></WINDOW>;

Startet den Timer **CLEARSCREEN** nach dem Befehl **<TIMER><START=CLEARSCREEN></TIMER>** einmalig und führt nach fünf Sekunden entsprechende Befehlsfolge aus.

Neue Klasse TIMER:

Befehle:

START={TimerName}	*- startet den Timer {TimerName}*
STOP={TimerName}	*- stoppt den Timer {TimerName}*
LIST	*- listet alle aktiven Timer auf*
RESET={TimerName}	*- setzt den Zähler von {TimerName} zurück*

Namen für Timer **{TimerName}** können beliebig verwendet werden. Es sind maximal 256 Timer möglich.

Beispiele:

<TIMER><START=IVT></TIMER

Startete den Timer Namens IVT, wie in der timer.csv definiert.

<TIMER><LIST></TIMER

Generiert eine Liste aller aktiven Timer. Wenn kein Timer aktiv, bleibt der Listeneintrag leer.

6.5 Counter Support

m..myhome ist ab der Version 1.0.91 unterstützt auch Counter. Counter werden wie Sysvars behandelt, werden jedoch nie gespeichert. Mit jedem Auslesen eines Counters wird der vorherige Wert um eine einstellbare Schrittweite verändert.

Dazu gibt es die passenden SYS-Befehle:

SETCOUNTER={Name}{,Startwert}{,Schrittweite}	initialisiert den Counter {Name}, {Startwert} und {Schrittweite} sind optional. Default ist 0 für {Startwert} und 1 für {Schrittweite}
SETCOUNTERSTEP={Name}{,Schrittweite}	ändert die Schrittweite eines bestehenden Counters {Name}, 0 für {Schrittweite} hält den Counter an.
GETCOUNTER={Name}	liefert den aktuellen Wert des Counters {Name}, ohne ihn zu verändern.
GETALLCOUNTERS	liefert die aktuellen Werte aller Counter, ohne diese zu verändern

In Befehlssequenzen kann mit **[{Name}]** auf den Counterwert zurückgegriffen werden (exakt wie bei den Sysvars, jedoch wird hier im Anschluß daran der jeweilige Counterwert um die Schrittweite verändert.

Beispiel für die Verwendung eines Counters um iTunes Titel als Laufschrift an den KNX zu übergeben.

In die init.myh den Befehl:

<SYS><SETCOUNTER=ITC></SYS>

eintragen. Dies bewirkt, dass der Counter mit dem Namen ITC bei Systemstart initialisiert wird. In die timer.csv kommt:

ITT;INTERVAL;1;<ITUNES><GETTITLE{%c[ITC],14}></ITUNES>;AUTOSTART

Dieser Eintrag definiert einen Timer „ITT" der in einem Intervall von einer Sekunde eine iTunes GETTITLE Abfrage auslöst, deren Rückgabewert durch den Parameter **%c** manipuliert wird. Der Parameter %c benötigt einen Index und eine Länge. Der Index wird von dem voherig definierten Counter (**[ITC]**) beschrieben und die Läge mit dem fixen Wert **14** belegt. Durch die Autostartfunktion wird der Counter automatisch bei Systemstart aktiviert.

Nun muss noch folgende EventServer Definition eingerichtet werden:

```
[CONFIG];;
TRIGGERIP;INTERNAL;
TRIGGERPORT;OUT
TRIGGERMODE;ASCII;
MATCHING;FULL;

[TRIGGERS];;
GETTITLE{\%}=\*|;<KNX><SETVALUE=1/5/5,\#></KNX>
```

Der TRIGGER parst auf GETTITLE, also den Namen des aktuell angespielten iTunes Titels. Der Wert der gematcht wurde, wird an die KNX Adresse 1/5/5 geschrieben. In der esf Typdefinition muss diese Adresse natürlich als 'Character String' (14 Byte)EIS 15 definiert sein.

Aufgrund der Abfrage, die über den Timer sekündlich initiiert wird, wird der Counter **[ITC]** immer um seine Schrittweite erhöht. Da bei der Initialisierung des Counters keine Schrittweite angegeben wurde ist der Defaultwert, also 1, angenommen worden. Durch den Parameter %c ist der Ausgabestring manipuliert worden. Es werden nur 14 Zeichen ab einem bestimmten Index ausgegeben. Der Index wird von dem Counter beschrieben, der bei jeder (sekündlichen) Abfrage um 1 erhöht wird. Die besonderen Eigenschaften des Parameters %c verhindern ein Überlauf. Siehe hierzu die Beschreibung zur Verwendung von Parametern in den *Formatierungsoptionen*

7. TABELLE key codes SYSTEM EVENTS

APPLESCRIPT Deutsche Apple USB-Tastatur, evtl. auch für andere Tastaturen gültig (alle Angaben ohne Gewähr):

key code 1 = "s"
key code 2 = "d"
key code 3 = "f"
key code 4 = "h"
key code 5 = "g"
key code 6 = "y"
key code 7 = "x"
key code 8 = "c"
key code 9 = "v"
key code 10 = "^"
key code 11 = "b"
key code 12 = "q"
key code 13 = "w"
key code 14 = "e"
key code 15 = "r"
key code 16 = "z"
key code 17 = "t"
key code 18 = "1"
key code 19 = "2"
key code 20 = "3"
key code 21 = "4"
key code 22 = "6
key code 23 = "5"
key code 24 = "´"
key code 25 = "9"
key code 26 = "7"
key code 27 = "ß"
key code 28 = "8"
key code 29 = "0"
key code 30 = "+"
key code 31 = "o"
key code 32 = "u"
key code 33 = "ü"
key code 34 = "i"
key code 35 = "p"
key code 36 = [Return]
key code 37 = "l"
key code 38 = "j"
key code 39 = "ä"
key code 40 = "k"
key code 41 = "ö"
key code 42 = "#
key code 43 = ","
key code 44 = "-"
key code 45 = "n"
key code 46 = "m"
key code 47 = "."
key code 48 = [Tab]
key code 49 = [Space]
key code 50 = "<"
key code 51 = [Bkw-Del]
key code 52 = [Return]
key code 55 = [Command]
key code 54 = [Command]
key code 53 = [Escape] esc
key code 54 = [Command]
key code 55 = [Command]
key code 56 = [Shift]
key code 57 = [ShiftLock]
key code 58 = [Alt]
key code 59 = [Ctrl] ctrl
key code 60 = [Shift]
key code 61 = [Alt]
key code 62 = [Ctrl] ctrl
key code 64
key code 65 = NUM [,]
key code 66 = [ArrowRight]
key code 67 = NUM [*]
key code 68
key code 69 = NUM [+]
key code 70 = [ArrowLeft]
key code 71 = NUM [X]
key code 72 = [ArrowDown]
key code 73
key code 74
key code 75 = NUM [/]
key code 76 = NUM [Enter]
key code 77 = [ArrowUp]
key code 78 = NUM [-]
key code 79
key code 80
key code 81 = NUM [=]
key code 82 = NUM [0]
key code 83 = NUM [1]
key code 84 = NUM [2]
key code 85 = NUM [3]
key code 86 = NUM [4]
key code 87 = NUM [5]
key code 88 = NUM [6]
key code 89 = NUM [7]
key code 90
key code 91 = NUM [8]
key code 92 = NUM [9]
key code 93
key code 94
key code 95
key code 96 = [F 5]
key code 97 = [F 6]
key code 98 = [F 7]
key code 99 = [F 3]
key code 101 = [F 9]
key code 100 = [F 8]
key code 102
key code 103 = [F11]
key code 104
key code 105 = [F13]
key code 106 = [F16]
key code 107 = [F14]
key code 108
key code 109 = [F10]
key code 110
key code 111 = [F12]
key code 112
key code 113 = [F15]
key code 114 = [Help]
key code 115 = [PgUp]
key code 116 = [Home]
key code 117 = [Fwd-Del]
key code 118 = [F 4]
key code 119 = [PgDwn]
key code 120 = [F 2]
key code 121 = [End]
key code 122 = [F 1]
key code 123 = [ArrowLeft]
key code 124 = [ArrowRight]
key code 125 = [ArrowDown]
key code 126 = [ArrowUp]
key code 127
key code 128 = "a"

(ab 129 beginnt es wieder von vorn: 129 = "s", 130 = d", usw.

8. Workarounds, Previews

Unter Workarounds verstehen wir Funktionalitäten, die sich in der Erprobung befinden. Wir möchten Ihnen diese Funktionen nicht vorenthalten und bieten Ihnen an, diese zu testen und bitten ggf. um Ihr Feedback. Bitte beachten Sie, dass kein Anspruch auf Evaluierung dieser Funktionen besteht. Es kann also sein, dass wir nachfolgend beschriebene Möglichkeiten nicht weiter verfolgen.

8.1. DVD-Copy-Window

Die Copy-Funktion erwartet im gui-Ordner eine Datei DVDCopy.csv mit folgendem Inhalt:

[CONFIG];

ACTIVE;YES	*aktiviert die Kopierfunktion, default: NO*
PATH; {Pfad zum DVD Sammelordner}	*Der Pfad muss bereits vorhanden sein, beachten Sie die korrekte Namensgebung der Ordner. Der im „Finder" sichtbare Ordner „Filme" heißt real „Movies" Ein Beispiel für den lokalen Pfad der Standard-Ordnerstruktur wäre z.B.:* ***/Users/mmh/Movies***

Funktionsweise:

Legt man eine Disk ein, wird zunächst geprüft, ob sich auf der Disk ein VIDEO_TS-Ordner befindet. Falls ja, wird geprüft ob in PATH bereits ein Ordner mit gleichem Namen existiert. Ist dies nicht der Fall, öffnet sich oben mittig ein kleines Fenster mit dem DVD-Namen. Ignoriert man dieses Fenster, schließt es sich nach 10 Sekunden automatisch. Klickt man auf "Copy", beginnt der Kopiervorgang und der Fortschrittsbalken wächst, anschließend wird die DVD ausgeworfen.

Klickt man (egal zu welchem Zeitpunkt) auf "Abort", wird der Kopiervorgang abgebrochen, der unvollständige, bereits erstellte Zielordner wieder gelöscht und die DVD ausgeworfen.

Die Funktion greift auf die OSX interne Kopierfunktion zurück. Zukünftig kann auch ein Pfad zu einem Shellscript angegeben werden. Hierüber können dann auch andere Kopierprogramme aufgerufen und evtl. notwendige Parameter übergeben werden.

Kopierschutz:

Mit dieser Funktion können keine kopiergeschützten Medien kopiert werden!

8.2. iTunes Remote

Apple hat mit der neuen iPod Touch und iPhone Geräte-Generation das „Tor" für native Softwareanwendungen Dritter geöffnet. Apple selbst hat die Software **„Remote"** (kostenlos zu laden über den iTunes Applikation Store) zur Verfügung gestellt. Apple Remote ermöglicht das komfortable Steuern von iTunes und AppleTV mit Zugriff auf AirTunes Lautsprechern von eben diesen Geräten.

Wir zeigen Ihnen in diesem Kapitel, wie sie als Anwender von **m..myhome** diese Steuerung optimieren und entsprechende Vorteile sinnvoll nutzen und umsetzen können.

Da **m..myhome** nicht direkt auf die Remote Anwendung zugreifen kann, machen wir uns die Leistungsmerkmale des Eventservers zu nutze. Hierüber wird der **PLAYERSTATUS** (ab Vers. 1.0.44) er iTunes Anwendung überwacht.

Sobald iTunes über Remote, lokal oder über Drittanwendungen angespielt wird, meldet **m..myhome** den Status über den Broadcast Port.

<ITUNES>GETPLAYERSTATE=PLAYING|OK</ITUNES>

Diese Meldung kann mittels folgender EventServer Definition ausgewertet werden:

[CONFIG];
TRIGGERIP;192.168.50.55 - *lokale IP Adresse*
TRIGGERPORT;1038 - *lokaler Broadcast Port*
TRIGGERMODE;ASCII - *empfängt einen ASCII Code*

[TRIGGERS];
<ITUNES>GETPLAYERSTATE=*|;iTunesRemoteEvent.myh;ONCHANGE

Die Trigger Definition bedeutet im Einzelnen:

<ITUNES>GETPLAYERSTATE=*|;

Der Eintrag „*****" parst den Wert einer eingehenden Zeichenkette zwischen „**<ITUNES>GETPLAYERSTATE=**" und dem Abschlusszeichen „**|**"

Bei Eintreffen des o.g. Status String wäre die Bedingung erfüllt und der Match wäre „**PLAYING**"

iTunesRemoteEvent.myh;

Ist der Name des Scriptfiles, das ausgeführt wird, wenn die Bedingung erfüllt wurde. In diesem Fall erstmal völlig unabhängig von dem Inhalt des „gematchten" Strings.

;ONCHANGE

Damit jedoch das Script nur ausgeführt wird, wenn sich neben der erfüllten Bedingung auch das Match geändert hat, wird der Parameter ONCHANGE verwendet.

Wie in dem Kapitel zu dem EventServer beschrieben, kann der Match als Übergabeparameter[ARG] für den Scriptaufruf verwendet werden.

Bedeutung der Scriptfiles

Wir haben aus diesem Grund das aufzurufende Script, wie folgt aufgebaut, welches Ihnen die Funktionalität verdeutlichen soll.

<SCRIPT><RUN=iTunesRemoteEvent_[ARG].myh></SCRIPT>

Das Script **iTunesRemoteEvent.myh** dient in diesem Fall dazu, ein weiteres Script mit Parameteraufruf anzutriggern. Da das Match mit dem Wert „PLAYING" gefüllt ist, erfolgt der Aufruf des Scriptfiles, sofern vorhanden, **iTunesRemoteEvent_PLAYING.myh**

Das Script **iTunesRemoteEvent_PLAYING.myh** beinhaltet somit die eigentliche Kommandosequenz, die ausgeführt werden soll, wenn der **iTUNES Status** auf **PLAYING** gewechselt hat.

<ITUNES><FRONT><SHOW><MUTEOFF></ITUNES><SYS><OPENBUTTONS=ITUNES></SYS><BROWSER><HIDE></BROWSER><TV><MUTEON><HIDE></TV><HIDE><DVD><MUTEON><PAUSE><HIDE></DVD><SCRIPT><RUN=MRS_KUECHE(SC).myh><ARG=0></SCRIPT><SYS><DISABLESS><MUTEOFF></SYS>

… als Beispiel.

Selbstverständlich kann für jede Statusausgabe ein entsprechendes SCRIPT mit entsprechenden Funktionen angelegt werden.

8.3. Nutzung der Squeeze Box als IR Empfänger zur Steuerung von m..myhome

Die Squeezebox verfügt über einen Infrarot-Empfänger. Dieser kann entsprechend ausgewertet werden um zB mit der Squeezebox Fernbedienung oder auch der kleinen Apple-IR-Remote m..myhome oder andere Geräte fernzusteuern!

So geht's:

Im Squeezecenter unter Einstellungen - Plugins - xPL-Schnittstelle die "Infrarot-Verarbeitung" auf "Raw" stellen und abspeichern. Die xpl .csv für die entsprechende Squeezebox um folgende Zeilen erweitern:

```
[SCHEMA=osd.basic];
OSDWRITE;COMMAND=write
OSDCLEAR;COMMAND=clear
OSDEXCLUSIVE;COMMAND=exclusive
OSDRELEASE;COMMAND=release
OSDTEXT;TEXT=\#
OSDROW;ROW=\#
OSDCOLUMN;COLUMN=\#
OSDDELAY;DELAY=\#

[WATCH=remote.basic];
keys;REMOTE_KEY
```

Eine .csv für den Eventserver anlegen, die die Infrarot-Kommandos in m..myhome-Befehle umsetzt (hier für die Apple-Remote):

```
[CONFIG];
TRIGGERIP;INTERNAL
TRIGGERPORT;BROADCAST
TRIGGERMODE;ASCII

[TRIGGERS];
REMOTE_KEY=77e150bf;<SYS><VOLUP=5></SYS>
REMOTE_KEY=77e130bf;<SYS><VOLDN=5></SYS>
REMOTE_KEY=77e160bf;<TV><CHUP></TV>
REMOTE_KEY=77e190bf;<TV><CHDN></TV>
REMOTE_KEY=77e1c0bf;<TV><FSOFF><FRONT><HIDE><MUTEON></TV>
REMOTE_KEY=77e1a0bf;<TV><ON><SHOW><FSON><MUTEOFF></TV>
GETCHNAME=\*|;<SLIMSERVER><OSDTEXT=\n\#><OSDDELAY=10></SLIMSERVER>
|VOL=\*|;<SLIMSERVER><OSDTEXT=\nVolume: \#><OSDDELAY=10></SLIMSERVER>
```

Die Klasse <SLIMSERVER> muß natürlich ggfs. Angepasst werden, je nachdem wie die Squeezebox Definition benannt wurde.

Nach Neustart von m..myhome, die Apple-Remote auf die Squeezebox richten und EyeTV per Netzwerk über Infrarot fernbedienen:

- **Links/Rechts -> Programme umschalten**
- **Hoch/Runter -> lauter/leiser**
- **Play -> TV an**
- **Menu -> TV aus**

Als Feedback wird auf der Squeezebox der Sender bzw. die Lautstärke für 10 Sekunden angezeigt - eigentlich ziemlich sinnfrei, aber um zu zeigen, daß das bidirektional geht, durchaus brauchbar.

8.4. Nützliche Shell Kommandos

Ist der Scriptclient nicht verfügbar können auch Befehle über die Shell ausgeführt werden. Ebenso ist es möglich, Logausgaben zu beobachten.

Öffnen Sie hierfür das Programm „Terminal" welches sich unter ./Programme/Dienstprogramme befindet.

Geben Sie folgenden Befehl ein:

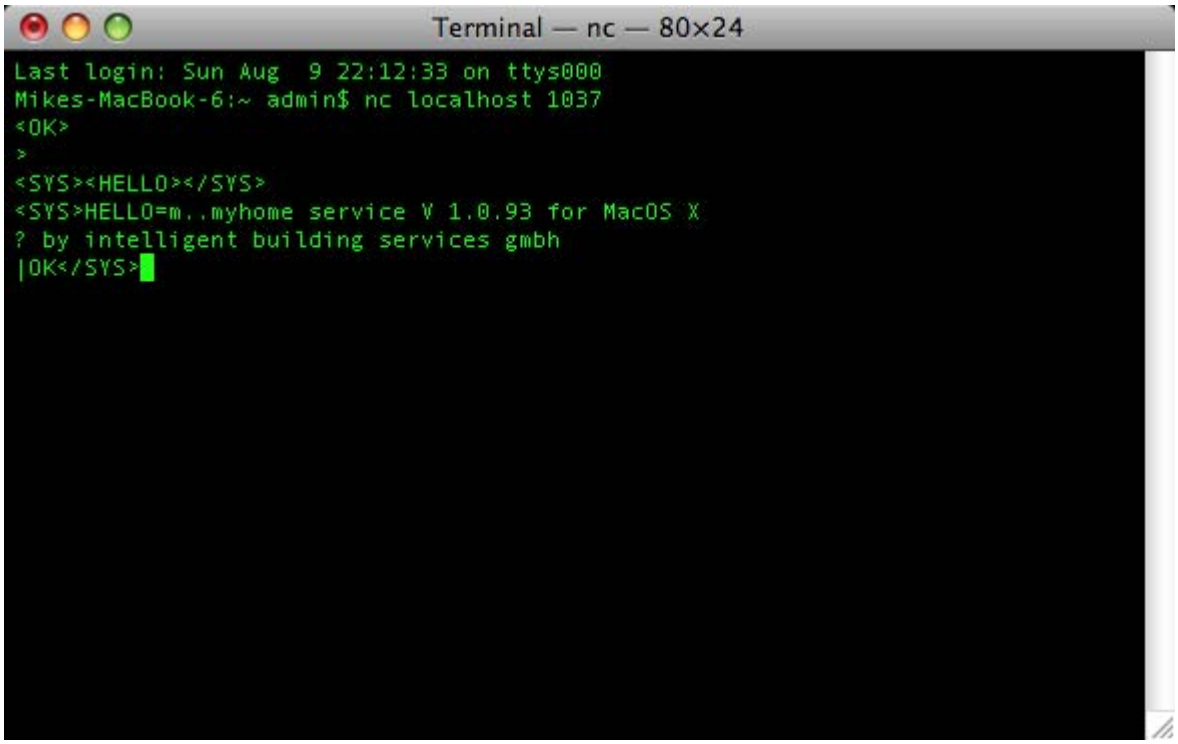
nc localhost 1037

Bei einer hergestellten Verbindung erhalten Sie die Meldung **<OK>**

Sie können nun m..myhome Befehle absetzen. Beispielsweise:

<SYS><HELLO></SYS>

Sie erhalten entsprechende Antwort:



Anstelle des Parameters „localhost" kann natürlich auch eine entfernte IP Adresse eingegeben werden. Vorraussetzung ist, dass auf dem Zielsystem der m..myhome Service gestartet ist.

Für Logausgaben verwenden Sie folgende Befehlsfolge:

nc localhost 1039

Mit diesem Befehl verbinden Sie sich auf den Logging-Port von m..myhome. Dies entspricht dem Logging Monitor im m..myhome Script Client.

8.5. Nützliche Shortcuts und Tastaturbefehle

Es ist immer wieder beeindruckend wieviele versteckte und unbekannte Tastenkürzel, Tastenkombinationen, Shortcuts es unter MAC OS X bzw. dem neuen Snow Leopard gibt um Anwendungen unabhängig vom Einsatz der Maus oder des Touchpads zu steuern. Im Folgenden entsteht eine Auflistung aller möglichen Shortcuts, Tastaturkombinationen unter MAC OS X/Snow Leopard und der darunter installierten Software um die tägliche Arbeitsweise an Ihrem Apple Computer noch effizienzter gestalten zu können. (Quelle: iFAQs.de)

iFAQs.de - Shortcuts für den Systemstart von MAC OS X/Snow Lepard:

[ALT] + [CMD] + O + F	Startet die Open Firmware Konsole beim Systemstart.
C	Der Rechner versucht zuerst vom internen CD-ROM Laufwerk zu booten
D	Der Rechner versucht zuerst von der internen Festplatte zu booten.
[ALT] + CMD] + P + R	Löscht das Paramter RAM.
[SHIFT] [ALT] [CMD] Backspace	Umgeht das aktuelle Startvolumen.
[ALT]	Öffnet bei neuern Rechner den Dialog zur Auswahl eines Startvolumens
[ALT]	Startvolumen bei Systemstart auswählen
[CMD] + V	Startet den Verbose Modus
linke Maustaste	CD beim Systemstart auswerfen
T	Rechner im Targetmodus starten
[SHIFT] + [ALT] + [CMD] + Delete	Umgeht das aktuelle Startvolumen und sucht nach einem alternativen Startvolumen.
N	Automatische Suche nach einem bootfähigen Netzlaufwerk
[SHIFT]	Systemstart im Safe Boot Modus
[CMD] + S	Startet den Single User Modus
D	Startet von der ersten Festplatten Partition
X	Startet OS X vom aktiven Startvolumen

iFAQs.de - Shortcuts für das Erstellen von Screenshots:

[SHIFT] + [CMD] + 3	Screenshot des gesamten Bildschirms
.	
[SHIFT] + [CTRL] + [CMD] + 3	Screenshot wird automatisch in die Zwischenablage gesichert.
[SHIFT] + [CMD] + 4	Auswahl mit dem Fadenkreuz welcher Bereich fotografiert werden soll.
[SHIFT] + [CMD] + 4 + Leertaste	Die Kamera fotografiert den aktuellen Bereich
[SHIFT] + **[CMD] + F4**	Das aktuelle Fenster wird fotografiert

8.6. Sonderzeichen (Mac/Win/VM-Ware)

Für das m..myhome Protokoll werden teilweise Sond3erzeichen benötigt, die nicht auf der Tastatur vorhanden sind. Ebenso ist die Tatsturbelegung bei Benutzung einer virtuellen Maschine nicht konform der Apple Tastatur. Nachfolgend zeigen wir die wichtigsten Tastaturbelegungen auf:

	MAC	WIN
@	ALT+L	ALT+Q
/	SHIFT+7	SHIFT+7
\	ALT+SHIFT+7	ALT+ß
[	ALT+5	ALT+8
]	ALT+6	ALT+9
|	ALT+7	ALT+<
{	ALT+8	ALT+7
}	ALT+9	ALT+0

9. Rechtlicher Hinweis

Das in dieser Software und diesem Handbuchbuch enthaltene Programmmaterial ist mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Die ibs übernimmt folglich keine Verantwortung und wird keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung der Software/Handbuches oder Teilen davon entsteht.

Das Werk einschließlich aller Teile ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung der ibs unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Auf den Zeitpunkt der Verfügbarkeit von angekündigter Funktionalitäten sowie angekündigter Erweiterungen besteht kein wie auch immer gearteter Anspruch, solange keine offizielle Freigabe der ibs intelligent building services gmbh vorliegt.

10. Versions Historie

mmh Release: 1.0.46 **(5. Erweiterung und Überarbeitung) Farbindex**

- Airport Support.
- Änderung CommandServer (CLIENTIP, MAPPREFIX)
- Erweitertes Logging
- Bugfix WINDOW-Klasse, träges Verhalten bei wiederholtem Aufruf einer URL

mmh Release: 1.0.45 **(4. Erweiterung und Überarbeitung) Farbindex**

- USB/RS232 Adapter Support f. CommandServer.
- …

mmh Release: 1.0.44 **(3. Erweiterung und Überarbeitung) Farbindex**

- BAOS Support für KNX/EIB.
- Autoskalier Funktionen in der WINDOWS Klasse erweitert
- Befehl WINSIZE für die Klassen BROWSER, TV, VLC, ITUNES
- Verwendung von Systemvariablen
- Broadcast Ausgabe um Statusmeldungen erweitert
- Wildcard-Match „*" im CommandServer-Reply wird jetzt unterstützt
- Erw. BUTTONS: Intervall- und Flankenfunktion eingebunden
- Neue Befehle in der SYS und DVD Klasse
- CommandServer Timeout nun auch in ms und je Befehl einstellbar
- CommandServer unterstützt nun auch das http Protokoll
- Aufruf der Eingabehilfe (Bildschirmtastatur)
- Namensänderung auf **m..myhome**
- Workaround zu Apple Remote
- Diverse Optimierungen durchgeführt (Buttonleisten, Scriptengine)

ScriptClient: 1.61 **(Anpassungen für mmh Version 1.0.44)**

- Neue Kommandos eingebunden
- Anzeige der Lizenzinformationen
- Kommandoliste nun auch in Englisch verfügbar

mmh Release: 1.0.38 **(2. Erweiterung und Überarbeitung) Farbindex**

- Windows, bei denen der Title gesetzt bzw. gelöscht werden, führen nicht mehr zum Absturz der GUI.
- Die RGW-Werte für die Textfarbe sind default auf 1/1/1, die Transparenz ist ebenfalls auf 1.
- werden XMIN und XMAX nicht gesetzt, werden die Default-Werte XMIN=0 und XMAX={Bildschirmbreite} verwendet.
- Durch die Umstellung auf Leo-Key-Events gingen teilweise Tastendrücke verloren

ScriptClient: 1.6 **(keine Änderung)**

mmh Release: 1.0.37. **(1. Erweiterung und Überarbeitung) Farbindex**

- generelle Überarbeitung und Vereinheitlichung der Kommandos
- Sprachengine eingebunden Klasse: SYS, Kommando: SAY
- xPL (Squeezebox) eingebunden, xPL Hub freigegeben (neues AddOn)
- EventServer freigegeben (neues AddOn)
- GUI Button Schnittstelle erweitert
- Erweiterte Start Scripts für „wake on Lan" und „Sleep"
- CommandServer überarbeitet
- iTunes GUI Ansteuerung gefixt
- Apple Script Fähigkeit eingebunden
- Workaround: DVD Kopierer

ScriptClient: 1.6 **(Anpassungen für mmh Version 1.0.37)**

- TCP/UDP Kommunikation überarbeitet
- Lizenzinformationen sichtbar
- Neue Kommandos eingebunden
- Abruf der xPL Klassen eingebunden

mmh Release: 1.0.21. **(Erstveröffentlichung)**
ScriptClient: 1.5 **(Erstveröffentlichung)**